This electronic thesis or dissertation has been downloaded from the King's Research Portal at https://kclpure.kcl.ac.uk/portal/

KING'S College LONDON

# Strategies for the Execution of Long-Term Continuous and Simultaneous Tasks in Grids

Haberland, Valeriia

*Awarding institution:*
King's College London

**The copyright of this thesis rests with the author and no quotation from it or information derived from it may be published without proper acknowledgement.**

END USER LICENCE AGREEMENT

**Unless another licence is stated on the immediately following page** this work is licensed

under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International

licence. https://creativecommons.org/licenses/by-nc-nd/4.0/

You are free to copy, distribute and transmit the work

Under the following conditions:

- Attribution: You must attribute the work in the manner specified by the author (but not in any way that suggests that they endorse you or your use of the work).
- Non Commercial: You may not use this work for commercial purposes.
- No Derivative Works - You may not alter, transform, or build upon this work.

Any of these conditions can be waived if you receive permission from the author. Your fair dealings and other rights are in no way affected by the above.

**Take down policy**

If you believe that this document breaches copyright please contact librarypure@kcl.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 05. May. 2023


# Strategies for the Execution of Long-Term Continuous and Simultaneous Tasks in Grids

by

Valeriia Haberland

A thesis submitted in partial fulfillment for the
degree of Doctor of Philosophy in Computer Science

in the
Department of Informatics
in the
School of Natural & Mathematical Sciences
King's College London

February 2015

# Abstract

Increasingly large amounts of computing resources are required to execute resource intensive, continuous and simultaneous tasks. For instance, automated monitoring of temperature within a building is necessary for maintaining comfortable conditions for people, and it has to be continuous and simultaneous for all rooms in the building. Such monitoring may function for months or even years. *Continuity* means that a task has to produce results in a real-time manner without significant interruptions, while *simultaneity* means that tasks have to be run at the same time because of data dependencies. Although a Grid environment has a large amount of computational resources, they might be scarce at times due to high demand and resources occasionally may fail. A Grid might be unable or unwilling to commit to providing clients' tasks with resources for long durations such as years. Therefore, each task will be interrupted sooner or later, and our goal is to reduce the durations and number of interruptions.

To find a mutually acceptable compromise, a client and Grid resource allocator (GRA) negotiate over time slots of resource utilisation. Assuming a client is not aware of resource availability changes, it can infer this information from the GRA's proposals. The resource availability is considered to change near-periodically over time, which can be utilised by a client. We developed a client's negotiation strategy, which can adapt to the tendencies in resource availability changes using fuzzy control rules. A client might become more generous towards the GRA, if there is a risk of resource exhaustion or the interruption (current or total) is too long. A client may also ask for a shorter task execution, if this execution ends around the maximum resource availability. In addition, a task re-allocation algorithm is introduced for inter-dependent tasks, when one task can donate its resources to another one.

# Acknowledgments

First of all, I am profoundly grateful to my supervisors Dr. Simon Miles and Professor Michael Luck for their tremendous support and encouragement throughout my PhD to deliver the best results, boosting my motivation and self-determination as a researcher. I have been extremely lucky with my supervisors who often sacrificed their spare time (e.g. weekends) to read my papers and thesis chapters, improving my research with their insightful advices. I also would like to thank Dr. Steffen Zschaler and Professor Peter McBurney for their valuable comments and inspiring ideas during my early PhD stages.

I am also tremendously grateful to King's College London for granting me with Graduate School Studentship and King's Overseas Research Studentship without which it would be impossible for me to study in the beautiful city of London and to complete this thesis eventually.

I would like to acknowledge my deep gratitude to all my friends and colleagues at the Department of Informatics in King's College London, who created a warm and friendly atmosphere for me to work here. I would never forget all great time we have spent together.

Finally, I would like to emphasise on my family's endless love and support during these years, which have been challenging to me in all different aspects of life. In particular, I am more than grateful to my father Grygoriy, my mother Olga, my sister Viktoriia and my grandmother Valeriia.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Introduction

Nowadays, a large number of applications are required to be run *continuously* or *near-continuously* over time in order to produce results in a real time manner. For example, these applications can monitor and / or control traffic congestion [1], processing the coordinates (other parameters) of vehicles in real time. There are also other examples of continuous tasks such as *continuous queries* [2], which process data streams continuously as soon as a new data has arrived without being issued repeatedly. For instance, they can be used in order to display complex live data from the multiple data sources (e.g. sensors), such as weather, traffic, etc. within a smart city scenario [3]. The process of *fetching* such complex linked data in real time is resource intensive as presented by Le-Phuoc et al. [3], i.e. it requires 32 quad-core nodes in order to fetch data from 320 data sources (10 sources per each node) for less than 250 seconds, while the smaller number of processors substantially increases this time. One more example of continuous tasks is presented in the work of McCormick et al. [4], where continuous planning is considered for military operations. Continuity is desirable for such applications as their results need to be up-to-date (in real time), but short interruptions are allowable [4], i.e. they do not affect noticeably a success of task execution, if not long enough for significant changes to happen e.g., the temperature does not rise or drop by more than one degree. The tasks which constitute such applications are therefore *continuous tasks* and it is desirable for them to be executed with no or short interruptions.

Another issue is that those tasks are usually resource intensive, processing streams of data in real time as discussed above. Therefore, they require large amounts of computational, storage and other resources, which can potentially be provided by a large-scale distributed resource sharing system such as a Grid [5]. Dedicated resources can also be deployed for these tasks by tasks' owners, which can be beneficial for those owners (e.g. organisations) as they do not need to compete for resources with other owners. However, if we assume a smart city scenario with potentially unlimited and dynamic number of continuous tasks e.g., a large number of vehicles may enter or leave a city continuously, it is impossible to deploy the additional amount of dedicated resources every time when they are needed due to technical and cost issues. There is also a question of resource distribution, i.e. such systems as Grids might be able to find resources as close to the data sources as possible [6]. A Cloud [7] is another environment as compared to a Grid which can potentially provide an unlimited amount of resources for continuous tasks, but its main idea is resource leasing rather than resource sharing, and the resource provider is a specific company for each Cloud. That is, the resources are allocated to tasks based on payments, which brings a problem of resource leasing cost. This problem becomes even more substantial, if continuous tasks are also infinite by nature [8] e.g., weather monitoring.

A Grid environment allows tasks to be executed, using the non-utilised resources, without leasing payments[1]. As long as a Grid's concept is resource sharing, the resources cannot be allocated for an unlimited time as they cannot be monopolised by a specific task. Hence, a resource allocation for such continuous tasks turns into obtaining the required resources for the longest possible durations with as short as possible interruptions. The problem of near-continuous task execution becomes even more sensitive to any interruption when the tasks are *inter-dependent* in terms of input data. For example, a task which monitors roads' intersection for possible traffic congestion may require the locations of moving vehicles from other tasks responsible for monitoring the adjoined roads. In this case, a substantial delay in data arrival from at least one task may result in a wrong decision for the whole controlling (or monitoring) process. Therefore, it is desirable for the inter-dependent continuous tasks to be run *simultaneously*, referring to their constant processing of data streams at the same time. However, if those continuous tasks allow short interruptions, then their near-simultaneous execution is also allowable.

[1]Some Grids can also be commercial, but an underlying Grid concept is a voluntarily resource sharing. Our work focuses only on the non-commercial Grids.

Although interruptions for these tasks are generally unavoidable due to the resource limitations, it is important for each task to be interrupted at times when resources are less scarce in order to be re-launched within a shorter time. An autonomous tasks' representative, i.e. a *client*, should also have a more significant role in a task allocation rather than just submitting a request as it is solely interested in effective task execution as opposed to a Grid, which might have other priorities (e.g. load balancing), and existing approaches do not necessarily account for this. Hence, a client should be able to affect a Grid's decision in respect of resource allocation, which can be achieved through negotiation [9]. In our work, a client has to negotiate with the Grid Resource Allocator (GRA), which represents abstractly a particular Grid. An exchange of requests and replies, where both sides concede in order to reach a mutually acceptable agreement, can be presented as a *negotiation process* [10].

Although the current work in Grid computing addresses different scenarios of task execution such as concurrent or ordered tasks, it largely overlooks continuous independent or inter-dependent long-term tasks' execution. That is, a continuity of task execution means that a success of the past task execution (e.g. the duration of uninterrupted executions) affects the worth of obtained resources in the future as well as a behaviour of the client towards the GRA. A client might become more conceding towards the GRA in order to obtain resources as quickly as possible, if the durations of its task's past interruptions have been too long. The research in the field of negotiation has been extensive, but it lacks negotiation strategies for clients who wish to execute such types of tasks in a Grid.

We discuss several specific Grid-related problems relevant to continuous long-term tasks in this chapter. These problems include resource scarcity and uncertainty in a Grid (see Section 1.2), unexpected resource failures or withdrawals and conflicting goals between the Grid schedulers and clients, resulting in interruptions to task execution (see Section 1.3), and data inter-dependencies among continuous tasks, where one task's failure affects the whole task tree (see Section 1.4). Our solutions to these problems are briefly discussed in Section 1.5, while Section 1.6 describes a thesis outline.

## 1.2 Resource Scarcity and Uncertainty

*Resource scarcity* in a Grid may occur due to intense competition among clients, when resources may potentially be available for one client's task but highly demanded by

other clients as well. Resources can also freely join or leave a Grid, or they can be shared only for a period of time while they are not in use by their respective owners. The level of *resource availability* in a Grid denotes the amount of resources that are not being used or assigned to tasks. A scarcity of resources means a mismatch in the amount of demanded and available resources, i.e. the amount of demanded resources is larger than the amount of available resources. If the demand on resources increases, its availability soon decreases and, as a result, the resources can be exhausted.

For example, resources can be more demanded during a day time and less demanded at night, they can also be more demanded during working days but less during weekends [11]. The higher the client demands in respect of resource amounts for its tasks, the harder it is to agree allocation of those resource amounts with a Grid, if they are highly demanded by other clients. In order to avoid resource exhaustion due to their allocation to other tasks, a client should be able to assess this risk and relax its demands if necessary. Therefore, it is important for a client at the time of resource request to know whether the resource availability is generally tending to decrease, increasing the risk of resource exhaustion. However, this information may be unavailable to a client, because of a dynamism of this information or a Grid's policy, which causes *uncertainty* for a client about resource availability. Then, how can a client estimate the tendencies in resource availability changes and adjust its demands as earlier as possible in order to run its tasks?

In our work, a Grid is represented abstractly by the GRA, i.e. we model a Grid in a general way, simulating some patterns of its behaviour in negotiation with a client e.g., the GRA becomes more greedy if resources become more scarce. We do not simulate a Grid resource topology and, as a result, the different overheads of task allocation, migration or communication between the GRA and a client. A client is not concerned with how the GRA is allocating resources, but rather how effectively its tasks are executed, depending on the resources obtained [12]. If information about resource availability is not directly available, a client may infer it from the GRA's replies in respect of its requests, i.e. during a negotiation process. The GRA is assumed to change its negotiation behaviour, when resource availability changes in a Grid. It behaves predictably for a client, if nothing changes in a Grid. We believe that this assumption is realistic, because a Grid's behaviour is not motivated by a monetary income, but by an intention to satisfy clients' requests and to utilise all resources. Therefore, when resources are not scarce, there is no need for the GRA to be greedy,

but when they are scarce the GRA has to become greedy in respect of each client as it intends to satisfy more requests.

Much work considers learning and predicting resource availability based mostly on substantial historical data (e.g. past negotiations), assuming resource availability fluctuations follow some deterministic patterns [13] e.g., a periodicity over time of the day. Some approaches do not require data from prior to negotiation, but then the initial assumptions can be far from the truth and, as a result, they require much trial and error before these patterns are identified [14]. One of our goals is to allow a client during a single negotiation to be able to estimate the risk of resource exhaustion from the early negotiation rounds, and to change its tactics according to the level of this risk, considering a lack of knowledge about resource availability.

## 1.3 Resource Contracting and Failure

A client and the GRA may have the *conflicting goals* in respect of resource allocation as the client aims to obtain better resources (e.g. the larger resource amounts), while the GRA might decide to allocate these resources for other clients rather than this one, following its own load balancing or resource scheduling policy. The reason for the GRA to be in a conflict with a client can be a high demand on resources, which means that other clients would have long waiting times if one client monopolises computing resources for a long time. The GRA might also be unsure about a computing resource capacity in the long term future [15]. The conflicting goals can be resolved through a negotiation process by reaching an agreement [9], which constitutes a *contract* between the GRA and a client in respect of task(s) execution. As we focus on continuous tasks for which time of execution is the most relevant issue, the GRA and a client negotiate about this issue, while we assume that the appropriate amount of computing resources will be automatically allocated when a time of execution is agreed. However, the amount of available computing resources is assumed to be behind the GRA's decision to allocate shorter or longer durations of task execution. Consequently, the lack of these resources or their uncertain future availability leads to the GRA allocating much shorter execution durations for the clients' tasks than a client initially aimed to obtain.

However, a client attempts to obtain resources for as long as possible in order to minimise possible interruptions and, consequently, its utility losses through negotiation with the GRA. The durations of execution for the continuous long-term tasks which

might be requested by a client can be measured in weeks, months, etc., and these durations are much longer than the typical durations of task execution in a Grid e.g., the majority of tasks are completed within 12 hours in AuverGrid [16]. A negotiation process between a client and the GRA involves a *strategising* from both sides, which means each negotiator choosing a specific course of action [17] according to its negotiation behaviour (i.e. generous or greedy), the external factors (e.g. resource availability), etc.

Traditionally, an agreement is reached when an opponent in negotiation proposes something which is better or equal to the negotiator's own proposal. However, some work also considers that a negotiator may agree to a slightly worse opponent's proposal under some conditions (e.g. tense competition) in order to reach an agreement faster [18,19]. The latter work does not focus on a continuous task model and, therefore, it does not consider that this decision may affect the conditions for future negotiations. In our work, a client does not just accept or offer a slightly worse proposal for itself in the current negotiation process, but the allocated time of task execution as a result of negotiation is aimed to ensure the client *better conditions* (e.g. higher resource availability) for future negotiations when this allocated time ends, considering the resource availability changes pseudo-periodically over time. A client can also become more generous in negotiation in order to reach an agreement faster, reducing a duration of interruption if this interruption is considered to be too long.

Our next goal is to address the problem of continuous task interruption by developing a negotiation strategy for a client, which allows a client to reduce a possibility and durations of interruptions, considering many factors e.g., the domain of the client application (e.g. the speed of change in temperature level in the office room), the client's level of risk-taking (e.g. a non-risky client intends always to reach an agreement faster), etc., which have not been addressed in the current literature.[2]

## 1.4 Data Arrival Delays

The problem of continuous task allocation and execution becomes even more complex if a continuous task depends on other tasks in terms of a data. That is, one task needs data from other task(s) to run, where each task sends its results as a stream to the other task(s) [2]. If at least one task is interrupted due to the end of its allocated time

[2]Note that these strategies can be applicable to both data independent and dependent tasks.

slot or a resource failure, then the other tasks which *directly* or *indirectly* depend on this task will be affected. A direct dependence means that the other task either directly depends on an interrupted task's data or directly sends its data to this task, not via other tasks as in the case of an indirect dependence. What happens if a task does not receive necessary data in time, given that it has to be run continuously? What happens if the receiving task has been interrupted and the data from its sending tasks is not processed any more?

Here, we focus on data arrival *delays* for inter-dependent tasks, because such delays may lead to an interruption of the data receiving tasks due to a large error in their estimations, causing an additional reason for task interruption, which is especially relevant for continuous tasks. If continuous tasks are inter-dependent, then an interruption of one task may lead to an interruption of other tasks, which is different from the case when tasks are considered to be data independent. Any interruption in those conditions causes a much larger utility loss for a client, and therefore a client might need an additional mechanism to obtain resources apart from a new resource negotiation with the GRA.

The consideration of *delays* in data arriving at one task from another task are introduced in the literature in different domains e.g., a data transfer delay through a network (e.g. wireless sensor networks [20]), a data delay due to a multi-agent coordination decision computation (e.g. web monitoring [21]), etc. A delay can also be a result of the interruption of a sending task, and this type of delay has received less attention in the literature, as the majority of continuous-task processing systems claim to be fault-tolerant, for example, migrating a task from a failed computing resource to another resource [22]. However, at the time when a task has to be re-allocated, there might be no available computing resources. Another approach can be *task replication* (i.e. running task replicas on the different computing resources), but this assumes the existence of redundant resources [23], which might not be available. A possible answer to this problem for a client is to re-allocate resources among its own tasks in the face of an inability to obtain new resources from the GRA. Note that a client is not authorised to re-allocate any resource by itself, but it may ask the GRA to perform this re-allocation as long as those resources have already been allocated. We also assume that the GRA might reduce the re-allocated resources (in our case the duration of task execution) in order to ensure some cost for a client, because every re-allocation leads to the GRA's own task migration cost [24]. We assume that it motivates the GRA to reduce the amount of re-allocated resources.

The current research lacks a comprehensive decision-making mechanism for clients to re-allocate their own tasks, especially where tasks are continuous. For example, the inter-dependent tasks' models lack implicit task dependencies, where a data sending task runs if its data receiving task runs as well which is vice versa in the case of explicit dependencies. That is, a sending task's data is virtually lost when it is not processed in real time in the case of its receiving task being interrupted. Such dependence is essential to take into consideration in the client's utility, because the sending task(s) of an interrupted receiving task are also technically interrupted as their data does not contribute into the client's system calculations any more. Our next thesis goal is to address the problem of continuous task interruption specifically for the case when tasks are data inter-dependent. As in the previous section, we aim to reduce the durations of interruptions, but not only by obtaining new resources from the GRA. Here, we also consider re-allocating clients' own tasks, taking into account data inter-dependencies among tasks.

## 1.5 Research Aims and Contributions

This thesis introduces novel negotiation strategies and a task re-allocation mechanism for a Grid client, aimed to protect its interests in negotiation with the GRA and to provide alternatives if a negotiation fails, in order to efficiently run continuous long-term tasks under conditions of uncertainty, high competition for resources and near-deterministic fluctuations of resource availability in a Grid system. As an illustrative example, this thesis follows a scenario of climate monitoring in a building, where the monitoring tasks process climate data (e.g. the temperature level) near-continuously for each room and near-simultaneously for all rooms in that building.

The contributions of this thesis are listed below.

*First*, to cope with resource availability fluctuations, reflected in the GRA's negotiation behaviour, under the uncertainty about those fluctuations, a *fuzzy logic-based controller* has been developed for a client. This controller allows a client to calculate its level of concession towards the GRA based on estimates of the GRA's negotiation parameters, which reflect its behaviour, computed on-line during negotiation. This contribution has been published as [25]:

V. Haberland, S. Miles, and M. Luck. Adaptive negotiation for resource intensive tasks in Grids. In K. Kersting and M. Toussaint, editors, *Proceedings of the 6th Starting AI Researchers' Symposium*, volume 241 of *Frontiers in Artificial Intelligence and Applications*, pages 125-136. IOS Press, 2012.

*Second*, to avoid resource exhaustion during negotiation, a unique *evaluation function* for a client has been developed, which indicates the risk of resource exhaustion every negotiation round. This risk is estimated by a client based on the overall tendencies in resource availability changes, inferred from the GRA's proposals. The client is able to vary the bounds of fuzzy sets on-line during negotiation based on this indication in order to calculate an appropriate level of concession.

This contribution has been initially presented at the Agent-based Complex Automated Negotiations workshop (ACAN) in the 12th International conference on Autonomous Agents and Multiagent Systems, St Paul, USA, 2013, and it is due to be published as [26]:

V. Haberland, S. Miles, and M. Luck. Adjustable fuzzy inference for adaptive grid resource negotiation. In K. Fujita, T. Ito, M. Zhang, and V. Robu, editors, *Next Frontier in Agent-based Complex Automated Negotiation*, volume 596 of Studies of Computational Intelligence. Springer Japan, 2015 (to appear).

*Third*, to facilitate the longer execution durations needed for near-continuous tasks, an algorithm has been developed, which allows a client to correct its negotiation outcome (i.e. a time slot to use Grid resources) with the GRA by offering a slightly shorter time slot of resource utilisation in order to start the next negotiation at the maximum of resource availability. This proposal is slightly worse for the client's current negotiation, but it allows for better negotiation conditions (i.e. higher resource availability) for the client in future negotiation.

*Fourth*, to facilitate shorter interruption durations for near-continuous tasks, a substantially extended evaluation function (compared to the second contribution) has been introduced which indicates when interruptions are too long to a client (in addition to the indication of the risk of resource exhaustion) in order to change its level of concession. Here, not only the current interruption is considered, but also a *total interruption* which includes all previous interruptions as it shows the success of task execution in the past.

The third and fourth contributions are also based on a novel continuous task model, and they are presented in a paper, which has been published as [27]:

V. Haberland, S. Miles, and M. Luck. Negotiation to execute continuous long-term tasks. In T. Schaub, G. Friedrich, and B. O'Sullivan, editors, *Proceedings of the 21st European Conference on Artificial Intelligence*, volume 263 of *Frontiers in Artificial Intelligence and Applications*, pages 1019-1020. IOS Press, 2014.

*Fifth*, to clarify the effects of a data arrival delay on continuous inter-dependent tasks, a model has been specified where tasks' dependencies are presented as a tree. Considering this model, a task re-allocation algorithm for a client has been invented, based on which a client may decide to stop a chosen task in order to allocate its resources for another task which has been interrupted for too long (a client still needs to negotiate this decision with the GRA as well).

*Finally*, to test the above mentioned decision-making mechanisms for a client in a realistic setting, a continuous Grid resource simulator has been developed. This simulator generates the observed in the literature dependencies of resource availability, demand on resources and total resource amount fluctuations over time. The realistic temperature changes over time in an office room have also been modelled, according to our illustrative scenario.

## 1.6 Thesis Outline

The rest of this thesis is organised as follows. Chapter 2 describes the background and the state-of-the-art research for continuous independent or inter-dependent tasks. It also discusses findings in Grid computing research, related to resource allocation, prediction and abstraction, where heterogeneous and large-scale computing resources are represented by an abstract scheduler for a client. Substantial attention is paid to the negotiation strategies for a client in this chapter.

Chapter 3 introduces a fuzzy-based controller for a client in order to calculate its level of concession towards the GRA, taking into account the risk of resource exhaustion, where the bounds of fuzzy sets are varied on-line during negotiation according to this risk. It also presents a comparative empirical evaluation for this client's decision mechanism in respect of other mechanisms such as the one which does not estimate the risk of

resource exhaustion (described in this chapter as well) or the one where a client has a full knowledge in respect of the GRA's negotiation parameters [28].

Chapter 4 describes a continuous task formal model, based on which a decision-making mechanism for a client is described which aims to facilitate longer task executions and shorter task interruptions. An empirical evaluation is made under different modelling conditions for the GRA.

Chapter 5 presents a continuous data inter-dependent tasks model, and an algorithm for a client to re-allocate a time slot of resource utilisation which has already been allocated to one of its tasks to an interrupted task, through negotiation with the GRA. This chapter also includes an empirical evaluation of this algorithm in various simulated Grid environments.

Chapter 6 describes a simulation of a Grid environment and the temperature level, based on real life dependencies observed in the literature. This chapter shows a comprehensive evaluation of the above described client decision-making mechanisms for a particular use case.

Chapter 7 concludes this thesis and describes our future work.

# Chapter 2

# Literature Review

## 2.1 Introduction

Having discussed our goals in Chapter 1, we present a review of background literature and the state-of-the-art research which investigates execution of continuous and simultaneous long-term tasks with a specific emphasis on running these tasks in a Grid environment where resources are allocated through autonomous contracting. This chapter provides term definitions and discusses research in respect of execution of these tasks (see Section 2.2); the Grid environment and, in particular, resource management (see Section 2.4), an automated contracting with a strong emphasis on negotiation strategies (see Section 2.5). Finally, Section 2.6 concludes this discussion by summarising open questions in these research areas.

## 2.2 Long-Term Tasks

In this section, we discuss the issues of execution of long-term tasks in a real-time manner. Much research work [6, 29–32] considers the necessity to process data streams in real time from the data sources e.g., radars, pollution sensors, etc. The tasks which process this data have to be run continuously (sometimes, simultaneously) to produce results in real time, i.e. the data is collected and processed fast enough in order to observe and/or affect any noticeable environmental changes. Here, we also assume that data is collected so often that it is impossible to allocate new resources for a

task every time when new data needs to be processed. For example, these tasks can perform a statistical analysis of this data [33] or they can perform filtering of this data according to client requirements [29], etc. Many researchers [8, 31, 32] state that these data streams are unbounded and infinite by nature. Therefore, they might require a large amount of resources to be processed for an undefined period of time. For instance, Ghanem et al. [30] state that current air pollution sensors can produce data every two seconds and it is approximately 8GB of data from each sensor per day. However, in reality, it is not possible to process an unbounded amount of data for the infinite time because the actual computing resources are bounded in terms of their capacity and operating time. Therefore, these tasks are more likely to be run for a long but finite time. The following sections discuss the different aspects of long-term continuous task execution in various domains (e.g. continuous planning [4]). However, our work considers an overall task characteristics, but it does not aim to contribute exclusively into the mentioned domains.

### 2.2.1 Continuity in Task Execution

The problem of continuous or pseudo-continuous task execution arises in diverse areas of life (e.g. plants, robots, air pollution monitoring). Some research seeks to find an optimal schedule for continuous processes (e.g. continuous mixing of feedstock) to utilise some resources (e.g. equipment) on plants [34], while other research addresses a dynamic area partitioning among a group of robots in order to monitor this area continuously in respect of some events of interest (e.g. gas leaks) [35]. The work of Chen et al. [6] proposes a *Grid-based Adaptive Execution on Streams (GATES)* to process distributed data streams in a real-time manner. For instance, this Grid system can be used to process tasks which can predict large earthquakes based on ground movements. Chen et al. also argue that a Grid environment is well-suited for the effective processing of data streams (i.e. continuous tasks). That is, the large-scale and distributed Grid might be able to allocate computing resources which are geographically close to the source of data that might reduce communication cost.

GATES aims to dynamically adjust the amount of arriving data with the amount of available computational, storage and communication resources to reach the best accuracy of results in real-time. Some other systems which process data streams, such as *DGClust* [8], *StarGlobe* [36], *Calder* [37] propose mechanisms to decrease computational and communication costs of processing data streams. For example, StarGlobe applies

filtering and parallel processing of data streams to decrease the level of resource load. In order to process continuous data stream, *continuous queries* are discussed in many approaches [2,38–41], and it is defined as the type of query issued once for a particular data-type and then runs continuously, updating a client with new results without being issued repeatedly. In other words, a continuous query is repeated over time to process continuous data streams.

Babu and Widom [38] state that processing of continuous data streams may require unbounded storage for the data traces. Therefore, they propose an architecture of continuous query processing, where an answer to a query can be sent to a permanent or temporary storage and it can later be deleted from a temporary storage. Newly arrived data can be also saved for some time and deleted later. In this way, the authors remove unnecessary data e.g., some outdated data, to avoid storage excess.

The idea of discriminative data storing is used in many other work such as Babcock et al. [42], Cammert et al. [43] and Mokbel and Aref [44]. Cammert et al. describe a *sliding windows* algorithm which determines the boundaries of the most recent data to be stored in-memory, and those boundaries can be reconsidered by a data stream management system over time. Babcock et al. propose a *load shedding* algorithm which reduces memory load by discarding some input data with some probability, estimated statistically in the way to preserve a satisfactory accuracy of results (e.g. an arrival stream rate is considered). Mokbel and Aref extend a load shedding algorithm which saves temporarily only that input data which is required by the active queries, and discards other input data. If this temporarily saved data is not required anymore, then it will be removed from a memory. Their algorithm also considers *spatial* and *temporal* characteristics of the data streams e.g., a stream of coordinates from a moving vehicle.

Sallam et al. [45] also address the problem of processing spatio-temporal continuous data streams, and discuss an issue of server scalability to process the large amount of data, which arrives at high rate online. They demonstrate that multiple collaborating servers, which are geographically distributed, can process a substantially larger number of queries than a single server by dividing the areas of responsibility for a moving object among themselves (in case those areas do not overlap). This solution also reflects relevance of Grid resources to process such continuous tasks as long as their resources are world wide distributed and ideally unlimited.

There is also work in other research fields which emphasises continuous time models for workflows (i.e. an ordered set of tasks [46]) or planning. In particular, Neophytou et

al. [47,48] state that a traditional workflow model, which works with a passive data and performs actions upon this data once it is invoked, does not reflect the current needs in processing continuous data streams for monitoring applications in science and business. Therefore, the authors propose their *continuous workflow* model, where the workflow is continuously active and it is responding on any events, connected with these data streams, in real time. Initially, Myers [49] discussed an idea of continuous planning in order to respond to any deviations of plan execution, which was also used in a more recent work of McCormick et al. [4] for air military mission execution. McCormick et al. demonstrate *Plan Execution Understanding Service* (PLEUX), which supports continuous plan maintenance and re-plans dynamically in a case of some deviations in plan execution. They also consider that a planned current or failed action might be re-planned and at that time it is considered to be interrupted, and this interruption is intended to be short. However, they do not offer any techniques to decrease or avoid such interruptions. Both McCormick et al. and Myers use a human support in planning decision-making, while the fully automated systems [50] also exist. Pettersson [50] defines the execution monitoring as a continuous task which is run in a real time in order to detect any abnormal events in the system (e.g. unexpected environment changes).

According to our discussion above, it has to be noted that research in many fields e.g., database management or planning, has shifted from investigating traditional static data (plans) towards continuously changing data (plans) in real time. Although all research discussed offers effective algorithmic solutions for their problem domains, they generally do not focus on such issues as computing resource scarcity and, as a result, data arrival delays and their impact on the system's stability and reliability. In addition, a client (an owner of a task) cannot effectively influence task execution and / or affect possible interruptions of these tasks.

### 2.2.2 Dependencies in Task Execution

Assume that continuous tasks, which are discussed in the previous section, depend on each other's data to produce adequate results in real-time. In other words, each task sends a data point to another task, while processing new input data. Hence, these tasks have to be run simultaneously in terms of processing each input data point as soon as it has arrived. Researchers attempt to solve similar problems in different ways. For example, Motwani et al. [2] focus on continuous queries (e.g. continuous tasks),

which process stream data from multiple sources. In this case, one query may process data not only from the initial source (e.g. sensors), but from another query as well. However, they do not discuss how delays or inaccuracies from one query might affect the results from another query, and whether the later query can be performed at all if it does not obtain some data, i.e. this work does not particularly focus on continuous tasks inter-dependencies in terms of a data. Other research by Barbieri et al. [39] proposes a query language *C-SPARQL* for continuous queries, which can obtain data from multiple sources and stream their output data to be used elsewhere. For example, such a query might count the number of cars which have passed a particular gate per some time unit. This query can also aggregate data from several sources by counting, adding, averaging, etc. the data instances over some time.

Much work suggests different platforms (engines) e.g., *Storm* [22], *Esper* [51], *StreamBase* [52], *Cyclops* [40], to execute continuous queries. These engines allow processing of data streams in real-time for a wide range of problems such as monitoring market dynamics, military exercises, web-site monitoring for e-commerce, etc. All those engines attempt to solve the problems of scalability, performance and memory usage in terms of execution of those data streams. However, the problem of tasks' inter-dependencies in terms of data exchange and how they can be run without some input data is not the focus of these engines. In the case of resource failure, it is assumed that tasks can be reassigned to another computational node [22], but the level of error (if applicable) because of interruption in data processing is not clear as well as whether such a node is always available. It also has to be noted that in an open and dynamic computational environment such as a Grid, where other clients also require resources, it might be difficult to re-allocate a task without affecting other clients' interests. As a result, the delays in task re-allocation should be expected and considered in terms of an engine's reliability and flexibility.

Much work also focuses on *linked data streams* such as [1, 3, 53] e.g., two data streams from sensors which monitor temperature and humidity in airport [3]. Other use cases, mentioned by Sequeda and Corcho [1] include traffic monitoring (control) where vehicle's speed and location are streaming in order to identify traffic conditions (e.g. traffic congestion) on a road. These examples show scenarios where continuous tasks can be linked in terms of a data. For example, tasks which process vehicle's speed and location, or tasks which process these data streams from the adjoining roads. It is realistic to assume that those tasks have to be run simultaneously in order to predict a congestion at any location of the monitored area.

Other work [21, 54, 55] focuses on accomplishing a high-level task by completing the number of *time-constrained possibly inter-dependent other tasks* e.g., gathering an information from the Web in order to offer appropriate products to the customers. Here, the tasks might have a hierarchical structure in this case e.g., the sub-tasks can be to check the web-sites $A$ and $B$ for a price on product $P$ and the higher-level task can be to compare those prices. Each task might have several sub-tasks as well as several higher-level tasks, which directly or indirectly depend on success in completing this task. Lesser et al. [21] offer their approach in planning mechanism, where a task can be completed with some degree of quality rather than just completed or not by a deadline. Some tasks can be executed simultaneously and some may need to wait for the results from other tasks and, as a result, their execution might be delayed.

Much research e.g., [24, 56–58], considers processing of inter-dependent tasks in Grid systems where dependencies are presented as a *directed acyclic graph*. In particular, Meriem and Belabbas [58] dynamically allocate tasks to resources, which arrive as a continuous stream over time. Dynamic allocation is meant to respond to any resource availability changes in a Grid, and resolve the problem of load-balancing at the runtime. Although all this work considers task dependencies in resource allocation, tasks are not considered to be repeated continuously in real-time. Nevertheless, this research describes relevant concepts which can be applicable to continuous tasks such as the *spare time* [24], which defines the maximal time of task execution before it affects the schedule of dependent tasks. Other work considers a *cyclic task graph* [59–61], where tasks are executed repeatedly over time and each task in a cycle obtains and sends data. Here, the cycles of task dependencies are represented in terms of data, instructions, etc.

In summary, all discussed research shows a wide range of applications of continuous interdependent tasks, which have to be run near-simultaneously as they have to produce the streams of data in real-time. It also points out the drawbacks of the existing research in respect of task interruption, especially repeated interruptions over time, which is either not explicitly considered at all or its influence on the client's system is unclear in the long-term. Generally, the research also lacks a description of some dependencies such as if a task sends a data to another task continuously and the task recipient stops, then the task sender's results cannot be processed in real-time, which is essential for continuous tasks.

## 2.3 Autonomous Agents

There are a lot of definitions of the term *"agent"* [62–66], because of the large variety of specific subclasses of agents based on different ideas, purposes, methods of reasoning about the world, etc. A widely used definition of an agent is proposed by Wooldridge, Jennings and Sycara [63,65,66], and this definition is also used in our work. According to them, the main characteristics of an agent are *autonomy*, *reactivity*, *pro-activeness* and *ability to socialize*. Autonomy means the ability of an agent to make a decision without direct commands from humans or other agents. Reactivity is the ability of an agent to react to the changes of its environment in real-time. Pro-activeness denotes the ability to initiate the analysis of their intentions and plan according to the events occurred in the environment. Social ability means to cooperate and communicate with other agents to perform their own or common tasks. The last three characteristics are sometimes combined into one definition called *flexibility* [63]. Wooldridge and Jennings [65,66] emphasise the social ability of the agents, arguing that agents should interact with each other to achieve the goal.

Müller [64] distinguishes between *hardware* and *software* agents. Hardware agents are physical entities, which deal with the physical world using hardware components such as sensors and effectors. However, they use software components too (for instance, a physical robot is a hardware agent with a software control system [67]). Software agents are computer programs, which sense, act on and reason about a software "world". Franklin and Graesser [62] point out the differences between software agents and non-agent programs. They insist that every software agent is represented by a program, but not every program represents an agent. The key differences lie in the *autonomy* of the agents and the way in which the agents interact with their environment. In comparison with non-agent programs, agents' actions as its output affects its input, i.e. the agent can change the type and number of input parameters, according to its actions. In contrast, non-agent program cannot change the set of input parameters, according to its output. For example, an agent tries to find an appropriate flight, considering such parameters as destination, date and price. In this case, a non-agent program tries to search for the flight using all three parameters at once. The agent can find the list of flights to the appropriate destination, then change its input parameter to date and start search by the date through the list of chosen flights. If the flight has not been found, it may propose to change the date, but it still can provide the list of flights according to the destination and price. Moreover, non-agent programs perform some tasks and

after that they wait for another call, while agents can reason about the environment continuously to perform appropriate actions in particular situations, because of their autonomous intelligent behaviour.

Traditionally, the description of *agent communication* actions is based on the theory of *speech acts*, which has two main ideas pointed out in work of Sbisa [68]. First, the difference should be considered between the meaning of a person's verbal *utterances*, i.e. the sense of what person says, and how these utterances are expressed such as request, order, informing, warning etc. Second, all types of utterances can be regarded as communicative actions, including the utterance of a statement. According to Allwood [69], Austin was the philosopher who proposed the distinctions between three speech acts known as *locutionary*, *illocutionary* and *perlocutionary* acts. A locutionary act means the understanding of the utterance (i.e. what a person says). For example, Ann requests the book and says "Give me this book", which means that Ann need to take this book. An illocutionary act denotes the underlying message of the utterance. For instance, "Give me this book" implies that Ann would like to read this book. A perlocutionary act is the real effect which was produced by the utterance. On the request "Give me this book", Ann's interlocutor gives her it back. Generally, the agents' abilities such as continuously responding to events in a dynamic environment and communicating or negotiating with other agents autonomously are essential to maintain continuous and simultaneous tasks' execution in a Grid.

## 2.4 Grid Systems

In general, a *Grid* is described by Krauter et al. [70, p.135] as *"a very large scale, generalized distributed NC system that can scale to Internet-size environments with machines distributed across multiple organisations and administrative domains",* where a *network computing (NC)* system is a distributed system that consists of a number of heterogeneous computational nodes, combined into a network, which consent to share their own resources with each other.

There are also other approaches, aside from Grid computing, which allow clients to use large amounts of computational and other types of resources. Some researchers [71,72] propose to use social networks in order to pool the large amounts of resources based on the trust relationships established among users. The users of social networks are

expected to be encouraged through payments or barter to share their personal computers. This approach potentially can pool a huge amounts of resources, but it also has stability and security concerns. Social networks' users are not necessarily aware of a proper computer maintenance such as virus protection, hardware and software maintenance, etc. Hence, this approach is not suitable at its current stage for interruption sensitive applications.

A Cloud environment [7] can also provide a potentially unlimited amounts of resources, but it has a greater degree of resource virtualisation with a more friendly user interface than a Grid. The concepts of Grid and Cloud computing are not mutually exclusive and generally aim to provide clients with the larger resource amounts than any single supercomputer or cluster. Cloud resources may also fail and they have their approximate peaks and lows of resource availability [7], while the durations of allocation are usually even shorter than in a Grid [16]. In our work, we focus on Grid computing rather than Cloud computing, but Grid technology can be also used as a basis for a Cloud [73].

### 2.4.1 Agents in a Grid

Generally speaking, Grid technologies [5,74] allow the coordinated sharing of dispersed resources based on highly distributed and dynamic *virtual organisations*. A virtual organisation denotes a collection of the different individual users and / or institutions which share their resources, according to some sharing policies (for example, who is eligible to share resources, what types of resources are shared, what terms are established for resource sharing) [5,75]. The above-mentioned issue of dynamism of virtual organisations (i.e. resources or services can become available or unavailable at any time) is raised in many works [18,76–78]. Rana and Moreau [79] state that agents (as described in Section 2.3) can respond to the dynamism of Grid resources, because of their adaptive nature. For example, agents can learn from their experience and, according to this knowledge, they can choose appropriate behaviour (a strategy) for a particular *situation*, a particular combination of specific agent's requirements, state of the environment and so on [80]. Moreover, Sim [18] states that the autonomy and intelligent behaviour of agents can be helpful for more sophisticated resource allocation (scheduling) in the Grid, as agents can employ different strategies to perform this allocation to the benefit of resource providers and resource consumers.

Foster et al. [81] point out the advantages of integrating Grid and agent technologies. According to these authors, Grid technology is mostly oriented to the development of a reliable and trustworthy platform (i.e. infrastructure, tools, interfaces) for high performance decentralised distributed computing, while agent technology focuses on autonomous and intelligent reasoning and comprises such social-oriented concepts as coordination, cooperation and negotiation. That is, agents can improve Grids by adding decentralised, autonomous, intelligent reasoning capabilities whereas Grids can provide agents with a reliable distributed platform on which agents can make their decisions.

Jonquet et al. [82] propose to follow a service-oriented approach of the Grid and multi-agent technologies integration. In this approach, *service* is indicated as an essential junction between Grid and multi-agent communities. A service is defined by Jonquet et al. [82, p.60] as *"an interface of a functionality (or capability)"* (e.g. an interface for such computational capabilities as the number of processors, operating memory etc.), that is compatible with the standards of a service-oriented architecture [83]. The general idea of such integration is to represent the groups of agents in terms of virtual organisations and agents' capabilities in terms of Grid services, and to couple the advantages of Grid and multi-agent systems.

### 2.4.2 Resource Availability Periodicity

Our work focuses on Grids where deterministic patterns in resource availability fluctuations can be identified. Hence, this section discusses research which investigates resource availability fluctuations and aims to determine some dependencies in those fluctuations over time. Nowadays, some research [13, 84–87] focuses on identifying deterministic patterns in Grid workload and resource utilisation in order to facilitate realistic Grid modelling. One of the initial works [88] to analyse host load fluctuations over time e.g., in clusters, shows that it is possible to predict load changes in future based on the historical record of those changes, which suggests the existence of some similarity in its fluctuations over time. Li et al. [87] statistically analyse the jobs' arrival times in data-intensive Grids, and identify such patterns as a pseudo-periodicity, the long-term dependencies, etc. The fluctuations of demand on resources, which is affected by the number and frequency of new jobs' arrival, might significantly contribute into shaping resource availability patterns. Moreover, the work of Iosup et al. [11] show some periodicity in their illustration of resource utilisation for the different Grid environments. Some of the evaluated Grid systems show time-of-day and day-of-week

tendencies in resource utilisation changes, where resources are generally more utilised during the working hours as well as working days rather than the non-working hours and weekends.

Andrzejak and Ceyran [84] propose methods to identify periodicity in resource demand, and they clearly demonstrate a periodic CPU utilisation over days of the week in their work. However, they also show less deterministic CPU utilisation over days for the different servers. Kondo et al. [89] discuss CPU utilisation in desktop grids, which is generally larger during weekdays than weekends. For example, their study suggests that CPUs are on average unavailable 19% of the time during working days and only 3% of the time during weekends, where availability means that a CPU should have more than 1% of free capacity to perform a task.

Feitelson [90] emphasises the importance of workload modelling in order to support a near-equal distribution of performance in a computational system. Di et al. [16] compare the workloads in a Grid e.g., AuverGrid [91], and Google Cloud. The authors found that the jobs' completion in the Grid generally takes longer than in Google Cloud as the Grid usually performs more computationally intensive tasks such as scientific calculations, while Google might perform such tasks as simple search. The frequency of new tasks' arrivals is also larger in Google than in a Grid. However, the average duration of task execution in AuverGrid is 7.2 hours, which is relatively short time in terms of the long-term tasks discussed in Section 2.2. Although this research provides an important analysis of the Grid workload for several Grid systems, it does show that the Grid tasks constitute comparatively short-term tasks, which finish execution when completed.

Extensive research [92–96] has been conducted in respect of predicting performance[1] in the Grid, as the changes of performance might have deterministic patterns over time. For example, Downey [93] uses statistical techniques to predict the waiting time for a task in a queue of other tasks to a multiprocessor system, aiming to obtain the desired amount of resources. An individual task benefits from this estimation when the decision has to be made whether the task should wait for the larger amount of resources or accept the system's offer. Although these predictions save a turnaround time for the clients' tasks, they exploit substantial information (e.g. the number of running tasks) about this system, which might not be available to a client.

[1] Performance may include a workload of processors in the system, the available RAM, throughput, etc.

Wolski et al. [96,97] propose the *Network Weather Service* (NWS) to make short-term forecasts related to resource performance based on its history. In this way, they estimate the execution performance for a task given particular resources. While Berman et al. [92] use the NWS to predict performance in their project of *Application Level Scheduling* (AppLes), they also take into consideration a possible variation of performance in their forecasts. High variation means that the predicted performance may drop significantly during a task's execution, making the resource worth less than one with lower variation. This work uses the history of change in resource performance (e.g. the change of throughput), while the client may not have access to such data.

Nudd et al. [95] create the *Performance Analysis and Characterisation Environment* (PACE) to evaluate and predict execution performance of applications in distributed systems. PACE aims to analyse the execution performance of applications before they run and to support them during the process of execution. Spooner et al. [98] use this environment in their task scheduling system *TITAN* in which they apply a genetic algorithm to choose the best schedules which satisfy the task requirements (e.g. the deadline of execution). However, the focus of this work is mostly on the task scheduling rather than the decision making for a client to obtain resources.

Other work [15,99,100] uses various statistical/mathematical techniques e.g., autoregression method (AR), to predict a CPU and/or network load in a Grid. Both Yuan et al. [100] and Akioka et al. [15] aim to predict computational workload far in advance, where Yuan et al. demonstrate load predictions for up to 250 minutes in advance (it could potentially work for longer durations) and Akioka et al. show quite accurate load forecasts for up to one week. Both works emphasise the complexity of such forecasts, which means that a resource scheduler may not be able to predict resource availability with adequate accuracy far in advance and, therefore, it would not be able to allocate tasks for long terms, because it should commit to each allocation.

### 2.4.3 Resource Abstraction

Clients (i.e. a program/agent which represents human requirements) are not usually interested in who the owner of a resource or the provider of a service is e.g., the particular organisation or organisations, according to the work of Buyya et al. [12]. However, clients are interested in price, speed, duration of resource utilisation, etc. That is, a resource's owner can be represented as an *abstract owner* for a client. For example,

when somebody uses a central heater in his/her flat, he/she usually does not need to be aware of what organisations produce raw materials, energy, tubes for heating and their prices for all of these resources, but he/she should be aware of the total price for heating. Therefore, the client negotiates with an abstract owner concerning requested attributes of resources/services through an appropriate interface. The interface could be an online payment form, issued by a local government who is responsible for heating in a particular region.

In terms of the Grid, this interface can be represented by a *meta-scheduler*, which sends a client's request to the *local schedulers* responsible for physical resource allocation [101]. A meta-scheduler might not only perform the role of an interface for a client, but it can be actually responsible for resource allocation e.g., in DAS-2 Grid system [102], where a meta-scheduler (e.g. *InterGrid Gateway*) obtains information from multiple resource providers about resource availability in terms of available time slots. This time slot includes such information as the number of available resources and their configuration as well as the time bounds when these resources are available.

One of the early works which proposed an abstraction of resources was the Legion meta-system [103]. This system is implemented in an object-oriented hierarchical manner where objects can be described as a high level abstraction of resources such as computational and storage resources (*Host Object* and *Vault Object* respectively). In other words, Host and Vault Objects are representatives of resources and they encapsulate such attributes of resources as operating memory, CPU type, disk space and so on. Another work [104, 105] also has a hierarchical structure of resource management, where the header agent keeps information about all "children" resources, but each resource is represented by an agent (see Section 2.3). In this case, an agent encapsulates physical resource capabilities. Shi et al. [106] also use agents to represent resources in their agent-based platform AGEGC for Grid computing, but their approach is decentralised, where local grid systems cooperate with each other in terms of information and resource sharing rather than storing their resource details on some meta-level agent.

An abstraction of physical resources in the Grid aims to simplify execution of high level applications, where a client does not need to know the specific physical structure of each computational resource. Xie et al. [101] propose such an abstraction for the case of multicluster Grids, where each cluster can be viewed by a client (application) as some *uniform* resource, i.e. each cluster is represented as a generic unit, stored in the *Virtual Cluster Pool* and accessed through a uniform interface. In other words,

the details of clusters (e.g. the amount of available resources) are hidden from the high level applications (i.e. clients and services).

Kee et al. [107,108] propose Grid middleware, *Virtual Grid*, which considers several resource abstractions: "bag", where heterogeneous resources with poor (i.e. "loose bag") or good (i.e. "tight bag") connections are aggregated or "cluster", where homogeneous resources with good connections are aggregated. Here, homogeneity means that each resource instance has nearly identical resource characteristics. Virtual Grid includes a descriptive language Virtual Grid Description Language (*vgDL*) for a client to express its requirements, where more terms can be expressed qualitatively rather than quantitatively which is more intuitive for human clients and abstractive for high level applications.

Adabala et al. [109] mention a number of advantages of abstraction in a Grid, where a physical resource (machine) can be presented as multiple virtual resources (machines) with various functionalities (e.g. an operating system, applications) at the same time, enabling a sharing of this physical resource by multiple clients with possibly different requirements. In particular, the authors insist on a higher level of virtualisation in their *In-VIGO* Grid computing system than in other work, where not only resources are virtualised, but also applications, data and networks are suggested to be virtualised. They also add several layers of virtualisation, where the virtualisation mentioned above constitutes only the first layer. The second layer composes these virtual components into services, and then services are presented as virtual interfaces for different access devices. Despite the advantage of virtualisation (abstraction) for Grid computing, Garbacki and Naik [110] also warn about its cost such as an additional memory is required to store the images of virtual machines and, therefore, they try to balance the usage of resource virtualisation in their work.

### 2.4.4 Resource Allocation and Task Re-allocation

A Grid is highly distributed and heterogeneous because it uses resources from different sources/ organisations that provide their own cost models and policies to handle these resources [12]. A resource can be a computational resource (for example, CPU time, number of processors), communication resource (communication channel bandwidth), storage resource (for example, disk space), resource slot (time for which other resources are allocated to a client) [102], etc. A client requires these resources from a Grid in order

to run its tasks and a Grid has to allocate them, satisfying task requirements [106]. Resource allocation in a Grid faces many challenges which include a high dynamism of resources requested by multiple clients, resource heterogeneity and distribution [111]. While the last two issues are something that clients cannot directly influence, the first issue significantly depends on resource consumption by clients' tasks. In addition, the level of resource scarcity or availability can be learned by a client, based on the resource proposals [112] or successful outcomes of negotiation [19]. In our work, a Grid does not share information about resources with clients, and this uncertainty for a client is discussed in the following chapter. This literature review is mostly concerned with resource dynamism issues in a Grid where this dynamism (i.e. resource availability changes) can be potentially observed or learned by a client. Some Grid systems also consider *task re-allocation* in order to balance resource load or other reasons [111].

#### 2.4.4.1 Conventional Resource Allocation

As a client may have more comprehensive information about its tasks than it is possible to pass to a Grid system, the client may significantly improve task execution in the Grid if it is allowed to affect in some way task execution, according to the Grid's policies. In this section, we discuss which well-known Grid resource management systems allow a client to affect task execution e.g., by relaxing task requirements [107], and to what extent. One such system is Condor [113], which distinguishes the concepts of matching and allocating resources to tasks. That is, a match between task requirements and resource advertisement does not mean a commitment between two parties. A client (*Customer Agent*) has to claim the matched resources from a resource owner (*Worker Agent*), and a resource owner should authorise resource allocation (e.g. the earlier advertised resources may no longer be available). Globus [114] offers a *resource specification language*, which facilitates an extensive negotiation between a client and Grid system, mostly motivated by the problem of adjusting task requirements with the resource availability during task execution. Some other well-known integrated Grid systems [115] such as NetSolve [116] and its evolved version NetSolve/D [117], and Ninf [118] and its evolved version Ninf-G [119], and a Grid project TeraGrid [120, 121] provide user-friendly interfaces for a client (a human user) to access the remote computational resources, but they do not actually focus on extending automatic client's involvement in the process of resource allocation and task execution. All of the work mentioned above lacks sophisticated decision-making for a client in order to set or amend task requirements. In approaches where a client is able to negotiate with a Grid

system (usually indirectly), there is an absence of sophisticated strategising to obtain better resources (e.g. a larger number of processors).

#### 2.4.4.2 Economy-based Resource Allocation

However, there are Grid resource allocation mechanisms where the clients (as well as a resource owner) strategise sophisticatedly in order to improve their utilities, and this approach is based on reaching a trade-off with a resource owner. It is generally known as an *economy-based* (or contract-based) approach, and it considers preferences of both resource owners and clients [12]. It allows clients to minimise their costs of resource utilisation and resource owners to maximise the profits of resource provision. A contract-based approach utilises known market models in negotiation such as auctioning or bargaining, and as a currency it may use real or virtual money, time slots of resource utilisation, etc. There are many contract-based models which have already been applied or can be applied for resource management in the Grid. The implementations of market models for distributed computing (i.e. solving problems of load-balancing, resource management/scheduling etc.) include such systems as Spawn [122], Mariposa [123] and Nimrod/G [124]. The Spawn system mostly focuses on effective allocation of idle computational resources based on a *sealed-bid second-price auction* [122, 125], whereas Nimrod/G performs more sophisticated resource management within a service-oriented Grid based on a *commodity market* model [124, 125] (Nimrod/G also supports several other economy-based models [125]). Mariposa [123] addresses such issues as the queries and storage management for distributed databases and it implements a *tender/contract net* economy-based model [125] in comparison with Spawn and Nimrod/G.

According to published work [125–128], auctions can be classified based on several characteristics. One such characteristic is whether the bidders can announce their bids openly or privately. A sealed-bid (e.g. Vickrey) auction means that participants are not aware of the bids of each other, as opposed to an open-cry (e.g. English) auction. Another characteristic is whether a client has to pay a price which it announced or another price. A second-price auction is one in which the winner of the auction (the one who proposes the highest price) pays the second highest price that was proposed. In the case that there is only one bidder, the resources are free because the second (highest) price is zero. This is different from a first-price auction, where the winner pays the price it has announced. There are also other characteristics such as the number of bidding rounds, the number of participants, the number of items (e.g. combinatorial

auctions [129–131]), etc. Although auctions provide a sophisticated competitive-based mechanism to allocate resources which can be compatible with the Grid systems, the clients usually receive only a response whether their bid was successful or not, but not a particular proposal. This limits the possibility for a client to learn its opponent's behaviour and, as a result, to use a more sophisticated strategy in negotiation.

However, a client can gather more information about its opponent's behaviour in a *bargaining* model [18,125,132] of negotiation, where participants exchange specific proposals over rounds. In this way, a client can increase its chances of obtaining desirable amounts of resources, time slots, etc. by applying more sophisticated strategies. This model is discussed in a detail in Section 2.5 with an emphasis on the client's strategising during negotiation.

#### 2.4.4.3 Task Re-allocation

Task re-allocation means that a task which has been allocated a resource is re-assigned to another resource, and this can be implemented by a Grid scheduler with an appropriate intelligence capacity and policy. For example, this might be necessary in order to balance the load on resources e.g., if one computational node is overloaded, then the task which causes this overload might be migrated to another resource [58]. Task re-allocation may be also decided by a Grid scheduler in the case when the currently allocated resources do not perform effectively [133]. There are a number of studies where task re-allocation proves to be effective for *advance resource reservation* in a Grid [134,135]. Advance resource reservation is when resources are allocated for a task to start execution at some point in the future. As a result of advance resource reservation, some space and time gaps might appear after new tasks request specific advance reservations. In this way, it becomes reasonable to re-allocate previously allocated tasks in order to avoid such gaps in the schedule, considering new tasks' requests.

However, Meriem and Belabbas [58] point out that dynamic task assignment might lead to overhead for a Grid system in terms of storing and monitoring the states of resources. Zhao and Sakellariou [24] also warn about a communication cost of task migration and computational cost of schedule re-computation. In this case, some work [136, 137] intends to avoid task re-allocation by estimating resource availability in advance, based on its past state transitions from being available to any other states, including an unavailable state. In this way, it is possible to predict whether a task will be successfully

completed within a particular time frame. However, if a task is near-continuous over time and it has to be executed for as long as possible without interruptions, then the possibility that such prediction will not be accurate increases, as discussed in Section 2.4.2.

In addition, all the work on task re-allocation mentioned above assumes that a Grid scheduler can find a resource for a task to migrate to if necessary, but not what happens if there are no available resources. A conclusion is that the discussed research is limited to a Grid scheduler's decision making without a significant impact from the side of a client, because a client, for example, may authorise Grid to stop one of its tasks in order to run others effectively. A Grid scheduler may not be able to decide this by itself as it has committed to task execution. Moreover, a client usually possesses more information about its tasks (e.g. task priorities, utilities, etc.) than a Grid scheduler and, therefore, it can make such decision potentially minimising its utility loss.

## 2.5 Automated Negotiation

According to Muthuchelvi et al. [138], negotiation is a process through which two or more participants attempt to achieve an agreement concerning desired resource attributes (for example, price, speed). Faratin et al. [17] refer to the work of Pruitt and describe negotiation as a process of relaxing demands between two or more participants until they find a common decision or have to consider other alternatives. Jennings et al. [10] state that negotiation is a process of reaching agreement which is acceptable for all participants in respect of a particular subject. Many researchers [9,17,18] argue that negotiation is an essential mechanism for resource management in a Grid, because both clients and resource owners are independent participants with different goals, requirements and preferences. That is, the mechanism is needed to find a common decision for both parties. Note that negotiation in this section refers to the bargaining model, where negotiators send each other proposals in order to reach a mutually acceptable agreement.

Sim [18] states that an automated negotiation can be used for resolving differences in negotiators' goals, preferences, etc. Agents abilities of autonomy, socialising, etc. can be useful for automated negotiation between a client and resource owner, where both of them are represented by agents. These abilities can also facilitate a collaboration among distributed resource owners which may have different policies and rules in respect of

resource provisioning [139]. Considering existing research [9, 17, 130, 131, 140, 141], a negotiation process can be classified according to the number of:

1. *issues* e.g., the time and cost of resource utilisation, the number of processors;
2. *participants* e.g., one-to-one (bilateral), one-to-many, many-to-one, many-to - many (multilateral);
3. *rounds*, i.e. one round denotes one step in negotiation, which usually includes a single exchange of proposals between negotiators.

Some research distinguishes between an issue (e.g. CPU time) and its attribute (e.g. the cost of CPU time). However, here, we only consider issues as the abstract subjects of negotiation, such as a CPU time or memory size. Every negotiation is based on three main components (this classification may vary in the literature):

1. a *protocol* which establishes the negotiation rules among participants in respect of the order of proposal exchange, possible reply messages (e.g. accept, reject, new proposal), the number of discussed issues, etc.
2. the *level of knowledge* in respect of the opponent(s) (i.e. an opposite negotiating side), competitor(s) (i.e. other agents which negotiate the same issues); environment (e.g. the resource availability in a Grid), etc.
3. a *strategy* which is used to generate a negotiator's proposal based on different conditions (e.g. the number of competitors).

In this section, we discuss research which has considered all these components to some extent with a major emphasis on the negotiation strategies for a client. In terms of negotiation strategising, negotiation approaches generally can be classified into three categories [10, 142]:

- *game theory* (e.g. [143–146]) - this approach aims to find an optimal strategy for a negotiator by searching a limited space of strategies, where both negotiating parties often possess significant knowledge about each other (such as their preferences, strategies, etc.);

- *heuristic* (e.g. [17, 25, 147–149]) - this approach tends to be less formal than a game theory approach in terms of negotiation models, and searches for a strategy which leads to a better outcome for a negotiator rather than an optimal one, where both negotiating parties often have an incomplete knowledge about each other;
- *argumentation* (e.g. [150–153]) - this approach adds more information on top of a proposal, allowing negotiators to explain their proposals, which may make it easier for a negotiator to understand the reasons behind its opponent's proposal.

An et al. [149] argue that a heuristic-based approach is the best for the dynamic and complex environments, where an optimal solution might be impossible to calculate for various reasons e.g., computational intractability. Thus, a heuristic-based approach seems to be more compatible with Grid settings than a game theory approach. Jennings et al. [10] state that an argumentation-based approach may improve negotiation by adding more clarity to the negotiators' proposals. However, it is unclear whether these additional arguments are useful in all situations, because they may cause an unnecessary computational overhead, resulting in longer negotiation. For Grid settings, where resource providers have different policies and tasks have different purposes and requirements, an argumentation framework will be either too generic (not helpful in negotiation) or too domain-oriented (helpful for only some tasks, but not widely applicable). Considering all discussed negotiation approaches, we mostly focus on a heuristic-based approach as it shows to be the most practical in the existing Grid settings.

### 2.5.1 Negotiation Protocols

A negotiation protocol defines the set of policies which determine possible interactions among participants [10]. Broadly, the protocols can be classified as bilateral or multilateral, depending on the number of participants. One of the well-known bilateral protocols is described by Rosenschein and Zlotkin [145, pp.40-41], the *monotonic concession protocol*. It starts when both negotiating parties submit their proposals at the same time, and if they overlap in terms of the negotiators' utilities, then an agreement is reached. If not, negotiation proceeds to the next round until an agreement is reached or no negotiators have conceded further. The overlapping of proposals in terms

of the negotiators' utilities means that at least one negotiator gains a better utility by accepting an opponent's proposal than clinging to its own.

The other widely implemented bilateral protocol is the *alternating offers protocol*, proposed by Rubinstein [154] and then adopted in much other work [9, 17, 25, 112]. This protocol assumes that negotiators exchange proposals in turns, which is different from a monotonic concession protocol. There are different versions of this protocol in respect of the options to reply and conditions to terminate negotiation. In general, the options to reply to the opponent's proposal for a heuristic-based approach are: (i) accept a proposal; (ii) reject a proposal and quit negotiation; (iii) reject a proposal and send a *counter-proposal* or a *critique* [155]. Here, a counter-proposal means a proposal in response to the opponent's proposal and it generally intends to be more appealing to an opponent than the previous negotiator's proposal. A critique contains a statement whether and/or why it likes or dislikes the whole or part of the opponent's proposal. A counter-proposal is much more common in this protocol than a critique as it offers a particular alternative solution rather than just a statement with its preferences. A negotiation process ends when one of the negotiators accepts its opponent's proposal, quits negotiation or a deadlock of negotiation is reached (e.g. the deadline of negotiation).

A multilateral protocol is often presented as an auction model e.g., [130, 156], or as multiple concurrent bilateral negotiations e.g., [157–159], coordinated by a central mechanism. Multilateral protocols may refer not only to a negotiation process with multiple opponents, but also to a negotiation process with multiple issues. For example, multiple concurrent negotiation processes may discuss multiple issues with one issue per process. Another example is a combinatorial auction [156], where a pack of issues is considered in each bid. There are also non-traditional protocols for multi-issue negotiations e.g., Klein et al. [160]. Here, the authors propose a negotiation protocol for multiple inter-dependent issues, using a mediator to generate a multi-issue proposal, which aims to be acceptable for both (many) negotiation parties. Each time all parties accept a proposal, the mediator changes one of the accepted proposal's values randomly and asks again up to a fixed number of times. If at least one party rejects a proposal, then the last accepted proposal is considered for mutation.

### 2.5.2 Uncertainty in Negotiation

The negotiators can be *competitive* or *cooperative* during a negotiation process [141]. Competitive behaviour denotes that the negotiating agents try to achieve a maximum utility for themselves. In such conditions, maximising of one's agent utility leads to minimising the utility of its opponent(s). Thus, the agents typically hide information such as their strategies, preferences and utility functions from other negotiating parties, which may give an advantage to its opponent(s) in negotiation. On the other hand, cooperative behaviour means that the negotiating agents try to achieve mutual benefit. Therefore, they often disclose their strategies, preferences and utility functions to other negotiating parties. Lawley et al. [141] state that cooperative behaviour might be more suitable for agents within the same organisation, while competitive behaviour might be more preferable for agents from distinct organisations. A Grid system represents a large number of distinct clients and resource providers, which are likely to exhibit competitive behaviour as they pursue conflicting goals or compete for the same resources. For example, a Grid scheduler may wish to allocate a shorter time slot to a client due to its scheduling or load balancing policy, while a client may wish to obtain a longer available time slot, which results in conflicting goals.

Information might also be inaccessible to a negotiator due to its dynamism and complexity and, as a result, computational intractability e.g., the increasing number of diverse client utility functions for a Grid scheduler. Therefore, in a real situation, the negotiating parties (e.g. a client and a Grid scheduler) are likely to have incomplete or no knowledge about each other and / or their environment. Some research [161–163] proposes learning the characteristics of the opponent during negotiation e.g., its preferences [161] or strategies / tactics [162, 164], or predicting the opponent's proposals using some negotiation history [165–167]. There are many learning techniques e.g., Bayesian [28, 162, 168, 169], reinforcement [112, 170, 171], and evolutionary [172] learning, and prediction techniques e.g., an auto-regression moving average [165], which are common in negotiation with incomplete knowledge.

However, some techniques e.g., Bayesian learning, often require prior domain knowledge or some initial beliefs about an opponent's behaviour. For example, a belief can be represented as a distribution of probabilities over some opponent characteristic values (e.g. its largest or smallest price), which might be difficult to obtain in a real life scenario. Narayanan and Jennings [162] indicate that reinforcement learning, which is

based on learning how to maximise a reward function, is ineffective in the case of non-stationary opponent's strategies, tactics, etc. The other issue for learning or prediction algorithms is that they usually need to have a substantial amount of data (event) points in order to produce an accurate result, while the resources in a Grid might be exhausted by that time. Hence, there is a necessity for techniques which can estimate such risk during negotiation in the early negotiation rounds without prior knowledge or time-consuming calculations.

While in negotiation with incomplete knowledge, a client has to operate with estimates rather than exact values, and so its decision-making mechanism is often presented as a *fuzzy-logic based controller* over a set of rules [17,173–176]. In fuzzy reasoning, a client makes a decision with some level of certainty based on some input data, which is similar to a human intuition and this data can be produced by experts, users, measurement devices, etc. For example, a person with £30,000 income p.a. can be considered "rich" with the certainty level 0.5 in a progressive country and with the certainty level 0.9 in a developing country, if this level is estimated in the interval $[0, 1]$, where one denotes an absolute certainty and zero denotes no certainty at all. That is, according to the average salary in the different countries, a person can be considered either "more" or "less" rich with some level of certainty for a particular individual. Fuzzy rules are usually represented as if-then statements [177] with the input and output *fuzzy sets* [178], defined by membership functions. Those membership functions can determine a client's numerical level of certainty for specific input data and a numerical output. Other fuzzy designs (e.g. a Sugeno-type fuzzy model [179]) offer linear equations or constants as an output of fuzzy rules rather than fuzzy sets. There is also a fuzzy design where an output can be a fuzzy set, linear equation or constant [180]. However, a Mamdani-type fuzzy model [177] with the output fuzzy sets is considered as the most intuitive and it is widely implemented in the different research areas.

All fuzzy models have to define fuzzy sets (at least as an input) and fuzzy rules, which are inter-dependent issues as the number of fuzzy sets determines the number of fuzzy rules. Lee [173] states that optimal fuzzy sets and rules are often produced by a heuristic trial and error method. For example, Sugeno and Yasukawa [181] are adjusting their fuzzy model to obtain a better performance e.g., adjusting the shape of membership functions and their number. Mamdani [177] uses reinforcements in order to produce the set of fuzzy rules, while Yin et al. [182] vary the shape of their membership functions, according to the user's feedback in equipment grid. Sim et al. [19] evolves the set of fuzzy rules, according to the successful negotiations, using the *genetic algorithms* [183].

Unfortunately, a client may not have a record of the successful negotiations so far, or this record may not be meaningful as circumstances may vary significantly in the dynamic environments. Liu et al. [184] state that the majority of fuzzy models still heavily depend on experts' knowledge and so are not fully automated. While the latter authors' fuzzy sets are initially designed by experts, they can automatically adjust to the changing environmental conditions using a genetic algorithm, which is also implemented in other work [172, 180].

There are many approaches in modelling fuzzy systems other than genetic algorithms e.g., *fuzzy neutral networks* [185], *unsupervised data-driven learning* [186], etc. Generally, many approaches aim to obtain optimal [187] or sub-optimal [172] fuzzy models in order to enhance performance in complex environments. However, there is a lack of focus on more flexible fuzzy models [182, 186] which respond actively to the client's goals but in a strategic way as they might be more suitable for the fast changing and complex environments such as Grids.

### 2.5.3 Negotiation Strategies

In Faratin et al.'s approach [17], a proposal and counter-proposal are produced by a particular *negotiation decision function*, considering only one criterion at a time such as time, resource or behaviour of a negotiator's opponent, and this function is called a *tactic*. However, a general course of actions is determined by a negotiation strategy. For example, if there is a risk of resource exhaustion, a client may decide to become more generous by changing its tactic (i.e. some coefficients of a respective function). Such a decision will be due to its strategy, as a client may choose other courses of action such as quitting a negotiation, waiting for the more favourable negotiation conditions, etc.

In general,

- a *time-dependent* tactic means that the amount that a negotiator concedes towards its opponent depends on time e.g., it may concede less at the beginning of negotiation, but more at the end of negotiation closer to its deadline;
- a *resource-dependent* tactic is similar to a time-dependent tactic, but the concession amount depends on resource availability e.g., a negotiator may concede more if there is a lack of resources;

- a *behaviour-dependent* tactic tries to imitate the behaviour of the opponent e.g., if a buyer's initial price was £10 then it proposes £15, a seller can reflect the difference in £5 between opponent's proposals and suggests £20 if its initial price was £25.

Here, the resource-dependent tactics might be impossible to apply if resource availability is unknown, but only its tendency of change can be guessed by a client. However, this criterion can be implicitly considered in the negotiator's decision-making mechanism. For example, a client might become more generous if there is a risk of resource exhaustion according to the tendency in resource availability changes. As for behaviour-dependent tactics, imitating an opponent might be risky if the negotiator is not aware of the reasons behind these proposals. For example, if the opponent behaves selfishly, i.e. it tries to increase its own utility in costs of the negotiator, the negotiator might behave in the same way towards its opponent [188]. However, the decrease in the Grid scheduler's proposed resource amount might be due to the resource scarcity, not its natural greediness, and selfish behaviour of the negotiator may lead to a negotiation failure. Although Hindriks et al. [188] consider modelling of the opponent's preferences, they do not focus on how the environment may affect those preferences e.g., resource availability.

Generally, the time-dependent and behaviour-dependent tactics proposed by Faratin et al. have evolved into a large class of strategies with these tactics as their basis. As the time-dependent strategies [189–191] naturally reflect many real life processes, we focus on this class of strategies. In addition, the time-dependent strategies can also implicitly or explicitly consider an opponent's behaviour or other conditions. For instance, some strategies try to incorporate not only time constraint, but also, for example, a changing market competition and opportunity (e.g. [19,192,193]), the behaviour of the negotiator's opponent (e.g. [17,161,192]), and even the time-independent tactics [194]. Here, we discuss the most influential work relevant to our research in the field of the time-dependent concession-based strategies predominantly for bilateral negotiation.

**Negotiation Decision Functions** Faratin et al. [17] offer two time-dependent tactics: *Boulware* (greedy) and *Conceder* (generous). In the case of the greedy tactic, a negotiator concedes less at the beginning of negotiation, but becomes more generous at the end of negotiation. In the case of the generous tactic, a negotiator concedes

more at the beginning of negotiation, but becomes less generous at the end of negotiation. Generally, a greedy negotiator does not concede a lot during negotiation, which is the opposite of a generous negotiator. Lang [148] considers a *Linear* tactic, where a negotiator makes the same amount of concession every round.

**Two-Phase Negotiation** Lang [148] proposes a multi-attribute negotiation model that is implemented in two phases: *distributive* and *integrative*. The distributive phase allows agents to maximise their utilities, i.e. to achieve the best payoffs for themselves. In this phase, all agents exhibit competitive behaviour, but it may lead to outcomes which are not *Pareto efficient*, because agents do not have enough information about the preferences and behaviour of their opponents. Pareto efficiency means that there is no better proposal which can increase the utility function of at least one agent without decreasing the utility function of any other agent [195]. In other words, there is no agent whose utility can be improved without losses to others. The resource allocation is considered to be optimal in the case of a Pareto efficient outcome.

In the case of the non-Pareto efficient resource allocation, Lang proposes an integrative phase that aims to find the mutual benefits for negotiating agents by slightly changing their payoffs from the distributive phase. If one agent finds more preferable values for its issues, it sends them as a new proposal to other agent(s). Then, that/those agent(s) can accept or decline this proposal and the negotiation process terminates or continues as in the distributive phase, using an alternating offers protocol. Alterations of the values of issues are performed according to a Gaussian distribution which denotes a higher probability for the small changes and a lower probability for the major changes [18,148].

**Market-Driven Agent** Sim [9, 18, 196] proposes a *Market-Driven Agent (MDA)* which considers a market dynamism in the Grid by implementing three functions: *opportunity*, *competition* and *time*. The MDA makes a decision about the amount of concessions by applying a combination of these three functions.

The opportunity function reflects the amount of pressure on a negotiator, depending on the number of possible *trading partners* (e.g. the number of resource providers for a client) and the differences in utilities between the MDA (a negotiator) and its trading partner produced by their proposals. Potentially, a higher number of alternatives may denote the lower pressure on a negotiator, because of the higher likelihood of finding a compromise. If the differences between proposals of the MDA and all trading partner(s)

are significantly large, then the probability of the agent reaching an agreement by its own deadline becomes low and a pressure intensifies. The competition function defines the probability that at least one trading partner considers the MDA its (their) most preferable partner in a particular round of negotiation. Consequently, the MDA has a higher probability of reaching an agreement if more (at least one) trading partner(s) rate its proposal as the most preferable. The time function reflects the time pressure on a negotiator, where a negotiator's concessions follow the same intuition as in Faratin et al. [17].

Furthermore, Sim describes a *"sit-and-wait"* time-dependent tactic [196]. This tactic denotes that a negotiator only concedes at its deadline, while waiting for its trading partner to concede. Sim also considers making reaching an agreement more likely by relaxing the values of negotiating issues. Typically, an agent accepts a proposal if the utility of its own counter-proposal is lower (or the same) than the utility of an opponent's proposal. According to Sim's relaxed terms negotiation protocol, an agent can accept a proposal which is slightly "worse" than its own proposal in the face of intensive pressure (e.g. a large competition).

**GENIUS** *Generic Environment for Negotiation with Intelligent multi-purpose Usage Simulation* (GENIUS) [197–199] is a platform for a bilateral multi-issue negotiation simulation between heterogeneous agents. It has a user interface, which allows a human user to choose the domain of negotiation, the negotiating agents with their strategies, preferences, etc. This platform also allows integration of new negotiation agents and it can compare an outcome of negotiation with some optimal strategies such as the ones which lead to a Pareto efficient outcome. This platform is also used to perform tournaments among agents, where the agents' performance can be contested in the different scenarios. Unfortunately, this platform is not designed to simulate negotiation processes for continuous task execution and, as a result, it performs multiple negotiations only for statistical reasons.

**Continuity in Negotiation** Generally, the reviewed work in negotiation does not consider client negotiation strategies for continuous tasks, where past negotiation success, resulting in a particular task performance, would significantly affect negotiations and task utility in the future. Here, we mean that an effectiveness of task execution (i.e. the durations of interruption) in the past directly affects its effectiveness of execution in the future (i.e. the worth of the obtained resources), not just as a historical

data which can be used for a learning in the future negotiations [19], but it has some formally expressed dependence over time (e.g. a client's utility). Some work considers the execution of continuous tasks, such as a continuous sweeping tasks for a group of robots [35], but it focuses on resources' (e.g. robots) coordination rather than the negotiation strategy for a client, which might wish to control its task execution in some way.

Here, we discuss the concept of continuity in a broader way rather than in respect of the tasks in order to show how this concept has been considered by other researchers. Some work [200] distinguishes the continuous values of negotiation issues rather than discrete ones e.g., a delivery time can be measured as 11.6 days rather than 11 days. Other work [201, 202] is concerned with one-to-many negotiation, which is not the focus of our work, where continuity of negotiation denotes that a negotiator can send proposals in a more flexible way and the new opponents may join in a negotiation process. The flexibility means that a negotiator does not need to wait for all proposals from multiple opponents to be received before sending them its counter-proposal, but it can send a counter-proposal to any opponent without waiting for others to arrive. This continuous-time mechanism also allows the new opponents to join a negotiation process in the middle.

## 2.6 Conclusions

Here, we summarise our discussion throughout this chapter. First, there is a huge demand for continuous task execution, either independent or inter-dependent, in different areas of science, industry, city infrastructure, etc.

Second, the existing algorithms, frameworks, etc. for these type of tasks lack the client's automatic and strategic involvement in the process of execution, while a client usually possesses more information about its tasks and it can make decisions to stop one task in order to run another task.

Third, a solution to the above mentioned problem can be a negotiation in order to reach a mutually acceptable agreement between the automatic representatives of a client and resource provider such as the autonomous agents, where multiple resource providers are usually abstracted for a client into a single meta-scheduler in a Grid, which is potentially suitable for processing these tasks.

Fourth, the existing work in automated negotiation tries to resolve the issues such as the resource availability dynamism in a Grid and an uncertainty in negotiation due to the different reasons e.g., a competitive behaviour of the negotiating agents, which hide their private negotiation characteristics. In addition, a client might be unaware of the resource availability due to its high dynamism or Grid policies. The researchers attempt to resolve the uncertainty and resource scarcity issues by predicting a Grid performance or learning an opponent's behaviour during negotiation. Generally, the work lacks mechanisms for a client to identify a risk of resource exhaustion in the early negotiation rounds. Some work is also not suitable for a dynamic Grid environment, or it requires a substantial historical data (training).

Fifth, the existing work in automated negotiation also significantly overlooks a continuity or near-continuity in task execution, where a negotiation success in the past should affect a negotiation process in the future, if a task has been interrupted due to a resource failure, resource scarcity, etc. As a result, the influence of interruption (especially, a cumulative one) on the client's system is unclear in the long-term future. This overlook is also reflected in the research about continuous interdependent tasks (not only in respect of negotiation). For example, the interdependent tasks' description lacks the implicit dependencies such as if one task which receives a data from another task stops, then that sender task has nowhere to send its results which will be eventually lost, and this effect should be reflected in the client's utility.

# Chapter 3

# Adaptive Negotiation for Resource Intensive Tasks

## 3.1 Introduction

Grids are dynamic, with the level of resources available fluctuating over time, and at different rates at each instant.[1] To obtain the resources needed to execute its tasks, a client may negotiate with a Grid Resource Allocator (GRA). In a Grid, there are two types of actor which influence the way resources are utilised: a client and a GRA. The client has an indirect influence on resource utilisation because it requires specific resources, while the GRA has a direct influence on resource utilisation because it allocates them. In our work, the client and the GRA are automated agents [65, 66] which act on behalf of organisations or individual humans in the Grid. Each client indicates the constrained requirements on resource utilisation e.g., the minimum required resource is 20 mega FLOPS. However, the client is not able to make a decision on how Grid resources are utilised e.g., it may require 30 mega FLOPS but it may obtain 20 mega FLOPS.

We consider that the GRA has full knowledge about resource availability in the Grid as opposed to a client, which does not have any precise information about the resource availability neither prior to or during a negotiation process. When multiple clients use

[1] The contributions, presented in this chapter, have been published (due to be published) in the papers [25, 26].

the Grid, resources can be exhausted during a client's negotiation because they have been allocated to other clients. Therefore, a client should be able to estimate such risk and change its negotiation tactics accordingly during negotiation, because a failure of negotiation is its worst outcome. However, a client may not be able to observe the resource availability accurately over time, either because of the high speed resource dynamism or the inaccessibility of this information due to the Grid's policy. Through observing the proposals of the GRA in negotiation, a client may make only estimates as to the speed and direction of the change in availability, because it is assumed that the GRA is more greedy when resources are more scarce.

In this chapter, we describe an adaptive negotiation strategy that allows a client to adjust its negotiation tactics not just to the change in resource availability at the current moment of time, but to the tendency for this to change over multiple negotiation rounds through a fuzzy control mechanism [25, 26]. The client does not have a precise knowledge about the resource availability changes, it makes its decisions in fuzzy terms [178], where the crisp input values are assigned some intuitive linguistic terms. For example, an increase in the resource availability of 50% compared to its previous value, can be considered a "large" or "medium" increase with some level of confidence. Then, fuzzy control rules operate with these linguistic terms in order to infer the change in the client's tactic e.g., if an increase is "large", then the client should become "more greedy" with some level of confidence in order to increase its utility. Our strategy aims to choose that tactic which maximises the expected amount of resources ultimately allocated to its tasks, considering the changes in resource availability and the risk of resource exhaustion during negotiation.

This chapter is structured as follows. Section 3.2 formalises a negotiation process between a client and the GRA, while Section 3.3 describes our adaptive negotiation strategy for a client. Sections 3.4 and 3.5 discuss evaluation results and conclusions respectively. Sections 3.2 and 3.3 include tables of notations, which are introduced in these sections (see Tables 3.1 and 3.3).

## 3.2 Formal Model

In this section, we describe a client's task specification, its utility and conditions of negotiation which include a negotiation protocol, the fundamental time-dependent tactics

for a client and the GRA [17] and their level of knowledge about each others' negotiation parameters. In the following chapters, this formal model is extended and/or modified.

### 3.2.1 Tasks and Proposals

As we have discussed in Chapter 2, the tasks which process data streams are resource intensive e.g., processing of air pollution data from sensors [30]. In this chapter, we do not specify the nature of a Grid resource e.g., CPU time, but we focus only on its abstract amount $r \geq 0$, which is presented as a non-negative real value.

#### 3.2.1.1 Task Description

In our work, a client has multiple tasks which have to be executed with specific resources. For example, the minimum resource required for a task can be the minimum acceptable computational performance to execute the task. Moreover, we distinguish *minimum* and *maximum* resource requirements. A minimum resource requirement must be satisfied to execute the task, while a maximum resource requirement has to be satisfied to execute the task in the best way for a client. We also assume that a client has different requirements for different tasks. Therefore, each task has different requirements in its specification.

*Notation* 3.1. A task $i \in \mathbb{N}$ describes a goal (e.g. the temperature level monitoring in a particular room) which is intended to be accomplished by an executable, associated with this task.

*Notation* 3.2. A specification $Spec_i$ of each task $i$ describes how the task's executable has to be run in the Grid, and it consists of the following:

1. The identifier of the task, $i$.
2. Minimum and maximum resource requirements $R_i^{min}$ and $R_i^{max}$ to run the executable.

$$Spec_i = \left(i, R_i^{min}, R_i^{max}\right),\ i \in \mathbb{N},\ R_i^{min} > 0,\ R_i^{max} > 0. \tag{3.1}$$

Each task $i$ also has private attributes which are known only to the client, but they are not available to the GRA. In this chapter, a client has only one private attribute, while the number of these attributes will increase in the following chapters.

*Notation* 3.3. Each task $i$ has a tuple $Task_i$ of private attributes, and this tuple (here, it is a singleton) consists of the time deadline $t^c_{dl_i}$ by which this task has to be launched, where the upper index "c" denotes that it is a client's deadline, i.e.

$$Task_i = \left(t^c_{dl_i}\right),\ i \in \mathbb{N},\ t^c_{dl_i} > 0. \tag{3.2}$$

The client's deadline $t^c_{dl}$ is unknown to the GRA, because the GRA might use this information in order to persuade a client for the larger concessions. The client submits all of its tasks at once to the Grid as one *job* (see Definition 3.1). Here, we assume that all tasks in one job are executed independently in terms of the input data, but they have common attributes. For example, the tasks can monitor temperature in the different rooms independently, but it is desirable for them to be run at the same time within one building. In this chapter, all tasks have the same time deadline $t^c_{dl}$ to be launched.

*Definition* 3.1. A job is a specification of a set of tasks whose executables have to be run by the specific deadline $t^c_{dl}$, i.e.

$$Job = \{Spec_1, ..., Spec_i, ..., Spec_N\},\ i \leq N,\ t^c_{dl_i} = t^c_{dl_j},\ i \neq j,\ i, j, N \in \mathbb{N}, \tag{3.3}$$

where $i$ is the identifier of the task in the job and $N$ is the total number of tasks in the job.

#### 3.2.1.2 Proposals

A *proposal* is a message sent by one negotiator in round $j$, where one round constitutes one exchange of proposals between negotiators, to another and it contains the proposed amount of resource for a particular task. That is,

*Definition* 3.2. A proposal $Pr_{i,j}$ denotes the amount of resource $\hat{r}_{i,j} \geq 0,\ \hat{r}_{i,j} \in \mathbb{R}$ which is offered to execute task $i$ in round $j$.

$$Pr_{i,j} = (i, \hat{r}_{i,j}),\ i, j \in \mathbb{N}. \tag{3.4}$$

A set of proposals denotes $Ps_j = \{Pr_{1,j}, ..., Pr_{i,j}, ..., Pr_{N,j}\}$ for all tasks in one job. Each of the negotiators send a set of proposals in each negotiation round $j$.

#### 3.2.1.3 Client's Utility Function

A utility function allows us to estimate the benefit gained by a client when its tasks are completed. The utility function describes the level of satisfaction of client requirements (see Notation 3.2).

*Definition* 3.3. The utility function defines the level of satisfaction of client requirements by mapping the obtained amount of resource $r_i$ for each task $i$ and the number of successful negotiation tasks $N_{suc}$ for which those amounts are allocated, to a real value $U_{Total}$ from the interval $[0, 1]$. That is,

$$U_{Total} : \{r_1, ..., r_i, ..., r_N\}, N_{suc} \rightarrow U_{Total} \in [0, 1], \ i, N_{suc}, N \in \mathbb{N}. \tag{3.5}$$

The client's utility function is a function of two variables $U_R$ and $U_T$. The variable $U_R$ measures the benefit for a client as the obtained resource amounts for all tasks in one job, while the variable $U_T$ measures this benefit as the number of tasks that are successful in obtaining resources $r_i \geq R_i^{min}$ in a single negotiation, where a duration of negotiation does not exceed $t_{dl}^c$. First, the variable $U_R$ is calculated as a mean of estimates $U_i^r$ over $N$ tasks, where $U_i^r$ shows the level of client's satisfaction in respect of the allocated amount of resource $r_i$ for task $i$. The estimate $U_i^r$ is equal to zero (its lowest value) if the task does not obtain its minimum acceptable amount of resource $R_i^{min}$, i.e. $r_i < R_i^{min}$. It is equal to one (i.e. its largest value) if the maximum amount of resource $R_i^{max}$ is allocated for task $i$, i.e. $r_i = R_i^{max}$. It is smaller than one but larger than zero, when the allocated resource amount $r_i$ is between its minimum and maximum resource requirements.

This estimate's value decreases linearly when a client concedes towards the GRA, starting negotiation from one by proposing $R_i^{max}$ and finishing it at least with $k_r^i \neq 0$ by proposing $R_i^{min}$ if a negotiation has been successful. The coefficient $k_r^i > 0$ is chosen by a client and it denotes the value of estimate when the minimum acceptable resource amount $R_i^{min}$ is allocated for task $i$, while the other coefficient is $k_r'^i = 1 - k_r^i$. Then, the estimate $U_i^r$ for task $i$ with respect to the allocated amount of resource $r_i$ is presented

in Formula (3.6).

$$U_i^r = \begin{cases} 0 & , r_i < R_i^{min}; \\ \dfrac{k_r'^i r_i + k_r^i R_i^{max} - R_i^{min}}{R_i^{max} - R_i^{min}} & , R_i^{min} \leq r_i < R_i^{max}; \\ 1 & , r_i = R_i^{max}. \end{cases} \tag{3.6}$$

Consequently, the variable $U_R$ is calculated as in Formula (3.7).

$$U_R = \frac{1}{N} \sum_{i=1}^{N} U_i^r. \tag{3.7}$$

Then, the variable $U_T$ comprises the number of successful tasks $N_{suc}$ divided on the total number of tasks $N$ as in Formula (3.8).

$$U_T = \frac{N_{suc}}{N}. \tag{3.8}$$

Finally, the total client's utility $U_{Total}$ is a geometric average between $U_R$ and $U_T$ as these variables represent two inter-influential factors (i.e. the amount of obtained resources and number of successful tasks), which are both crucial for a client. For example, if a client obtains a large amount of resources but only for a few tasks, $U_R$ will be devalued by the value of $U_T$ and vice versa.

$$U_{Total} = \sqrt{U_R \times U_T}. \tag{3.9}$$

The value of $U_{Total}$ is equal to zero, if no client tasks obtain any amounts of resource which satisfy the minimum amount of resource by the end of negotiation. The value of $U_{Total}$ is larger than zero, if all or some of tasks obtain the amounts of resource which satisfy the minimum requirements by $t_{dl}^c$. The value of $U_{Total}$ is equal to one, if all client tasks obtain a maximum amounts of resource $r_i = R_i^{max}$ by $t_{dl}^c$.

### 3.2.2 Negotiation Mechanism

This section describes the components of the negotiation mechanism which is implemented in our work and presented in Figure 3.1. This mechanism comprises the GRA's and the client's negotiation strategies (see Section 3.2.2.3), a negotiation protocol (see

Section 3.2.2.1) and the level of uncertainty in the Grid (see Section 3.2.2.2). In our work, the participants of negotiation are a client and the GRA (*Client* and *GRA* in Figure 3.1). That is, a client intends to obtain at least the minimum acceptable amounts of resources and at most the maximum amounts of resources for its tasks through negotiation with the GRA. The GRA intends to satisfy at least the client minimum requirements (i.e. $R_i^{min}$) and at most the client maximum requirements (i.e. $R_i^{opt}$). We consider only that case when the GRA has potentially enough resources to satisfy the client minimum requirements and it may have potentially enough resources to satisfy the client maximum requirements. However, these resources are also required by other clients in the Grid, and the GRA may choose to provide those resources to them. We consider a one-to-one negotiation (*bilateral*), i.e. a client with the GRA, while multiple clients are considered in our model implicitly. In other words, we assume that the GRA can also negotiate with other clients at the same time, but we do not focus on its policy to coordinate those negotiations. For example, the amount of available resources in the Grid may change, because other clients obtain or release these resources.

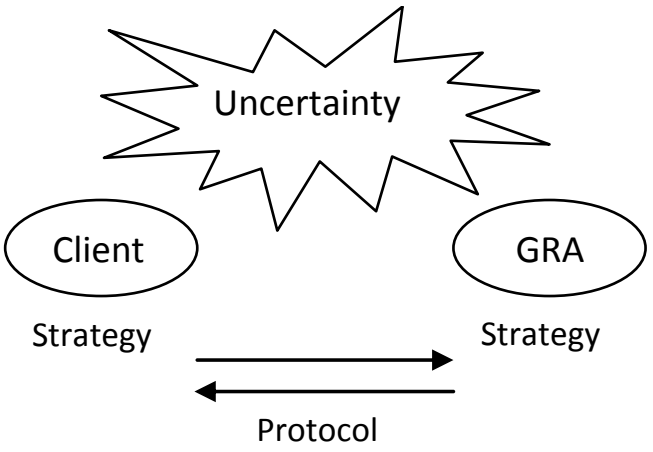



FIGURE 3.1: Negotiation mechanism

We assume that the GRA and clients have a limited amount of information about each other, i.e. they do not know utilities, strategies, preferences of each other, which leads to an uncertainty in negotiation (see Figure 3.1). This uncertainty might be caused by the difficulties to exchange or process the necessary information because of its size, complexity and / or heterogeneity (e.g. utilities). This uncertainty might also be caused by the intention of participants to increase their utilities, hiding private information from its opponent (e.g. strategies). For example, a client intends to obtain as better

as possible resources for its tasks, while the GRA might aim to be fair to all clients and to balance its resource utilitsation, which could be a motivation to offer the lesser amounts of resources to a client.

The negotiation mechanism needs to specify a negotiation *protocol* which establishes the rules of negotiation such as how participants should exchange their offers (e.g. a client and the GRA might submit their offers in turns or at the same time), how participants can reply on the offer of its opponent (e.g. a client might accept an offer of the GRA or reject it), etc., which is described in the next section. Moreover, each participant has its own preferences, aspirations, motivations in negotiation which are expressed in its *strategy*. For example, Schwind et al. [130] distinguish an impatient participant which has to bid a high price because it needs resources urgently. Another example is that a participant might make larger concessions for less important issues and smaller concessions for more important issues in Lang [148]. That is, the strategies of the client and the GRA define how much they have to concede in which round of negotiation, when they have to start, continue or quit negotiation, etc. The next two sections describe the negotiation protocol and the level of uncertainty in the Grid.

#### 3.2.2.1 Negotiation Protocol

In our current work, we adopt the well-known *alternating proposals protocol* [148, 149, 154] in which the pair of negotiators exchange proposals in turns. We do not consider a temporal or computational cost of decision making for a client or the GRA in our modelling. One proposals' generation for both parties and exchange of those proposals constitutes one negotiation round. Each negotiator may *accept* the opponent's proposal, *generate* the counter-proposal or *reject* the opponent's proposal without generating a counter-proposal. That is, the last option means that the negotiator quits negotiation. Figure 3.2 depicts the flow chart of the alternating proposals protocol implemented in our work. First, the client sends the message $SUBMIT$ which contains its job to the Grid (see Definition 3.1). Then the GRA replies with the message $PROPOSAL$ which contains the description of the proposed resources to execute tasks, specified in the job (i.e. a proposal which is defined in Definition 3.2). Then the client has three options to answer, i.e.

- Send the $PROPOSAL$ message which contains its own proposal, if the client is not satisfied with the GRA's proposal and it is not willing to quit negotiation.

- Send the $ACCEPT$ message which contains a token "accept", if the client is satisfied with the GRA's proposal and it is willing to accept it.
- Send the $REJECT$ message which contains a token "reject", if the client is not satisfied with the GRA's proposal and it is willing to quit negotiation.

In the same way, the GRA has the same three options to reply on the client's proposal. The negotiation terminates with:

- *Success* if the client or the GRA sends the $ACCEPT$ message to its opponent. This message means that the negotiating parties reached a mutually acceptable agreement.
- *Failure* if the client or the GRA sends the $REJECT$ message to its opponent. This message means that the negotiating parties did not reach a mutually acceptable agreement.

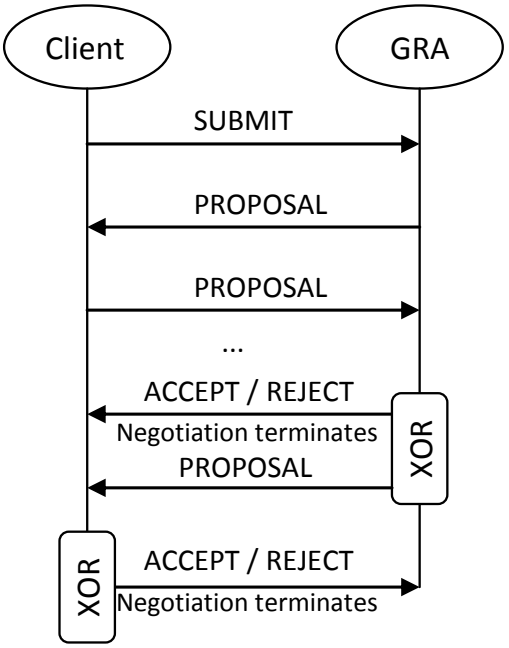



FIGURE 3.2: Alternating proposals protocol

We assume that negotiation can be terminated with a failure if:

- the negotiation deadline of at least one negotiator is reached and this negotiator does not agree with the last proposal of its opponent;
- the computational resources are exhausted and the GRA cannot make an offer to the client.

#### 3.2.2.2 Uncertainty in the Grid

The important part of negotiation is the level of participant's awareness about the subject of discussion (e.g. the technical characteristics of a car) and about its opponent (e.g. preferences, utility function). Fatima et al. [190] discusses the cases of the complete and incomplete awareness of participants about each other (e.g. utility functions). The authors also distinguish between the *symmetric* and *asymmetric* uncertainty of participants about each other's negotiation parameters or environment. Consequently, the symmetric uncertainty denotes that both participants are not aware of the negotiation parameters of each other, while the asymmetric uncertainty denotes that only one of participants is not aware of the opponent's negotiation parameters. In our work, a client has incomplete and asymmetric information about the GRA and Grid resources. For example, the GRA is aware of the amount of available resources in the Grid, but this information is not available to a client. A client discloses its minimum and maximum requirements to the GRA (see Notation 3.2), because a client requests resources while the GRA manages them.

**Uncertainty for a Client** We assume that a client does not have any information about the resource availability in the Grid, because of the GRA's policy or resource intense dynamism. A client has to rely on the truthfulness of the GRA when the GRA promises to execute client tasks with specific resources. In our work, we assume that the GRA cannot stop a client's task or re-allocate it unless a client asks the GRA to do it. Therefore, if the GRA promises to execute client tasks with certain resources, this allocation cannot be changed during the running of a task.

We assume that a client is not aware of the GRA's:

- Utility function e.g., the GRA's preferences among clients.
- Negotiation tactic, i.e. the level of greediness of the GRA in respect to the client.

- Reservation resource, i.e. the maximum possible resource that the GRA can offer to the client.
- Deadline of negotiation.

We assume that if there are potentially enough resources to satisfy client maximal resource requirements, which are not significantly demanded by other clients, the reservation value of the GRA is equal to the client's maximum amount of resource. However, if the resources are scarce, then the GRA's reservation value is likely to be smaller than the client's maximum one.

We also consider that the client knows that the GRA becomes more generous when the larger amount of resources is available and otherwise. In other words, when the resources are less scarce, the GRA intends to satisfy clients' maximum requirements. When the resources are more scarce, the GRA intends to satisfy clients' minimum requirements. The deadline of negotiation and reservation value of the GRA are not available to the client, because the GRA hides them. For instance, the GRA intends to reward resources to the more compromising clients in the case of resource scarcity. If the client is aware of the reservation value and the deadline of negotiation of the GRA, it could just wait when this deadline is reached and then it may obtain the reservation value of the GRA. In this case, the GRA might not be able to encourage clients for compromising.

We assume that a client does not have any certain information about other clients in a Grid, because a Grid does not disclose this information. In this way, the number of clients, their proposals, the number of accepted and rejected proposals are not disclosed. The Grid keeps this information confidential for all clients, because all of them are considered to be competitors. Therefore, the clients prefer to hide their utilities and requirements from other clients. However, the client knows that it has competitors for resources and that they may influence on:

- The reservation value of the GRA, i.e. the reservation amount of resource may change, because other clients release or obtain Grid resources.
- The level of greediness of the GRA, i.e. this level also depends on the resource availability.

When resources become more scarce (other clients consume resources), the GRA's reservation amount of resource decreases and its level of greediness drops. Consequently, when resources become less scarce (other clients release resources), the GRA's reservation amount of resource increases and its level of greediness rises.

**Uncertainty for the GRA** We assume that all resource providers in a Grid are represented by an abstract GRA for all clients. The GRA is aware of the resource availability in the Grid, the demand on resources, the resource technical characteristics and Grid topology.

We assume that the GRA is not aware of the client's:

- Utility function e.g., client's preferences.
- Negotiation tactic, i.e. the level of greediness of the client in respect to the GRA.
- Deadline of negotiation.

We assume that the client hides task private attributes (see Notation 3.3) and negotiation parameters, because their disclosure may decrease its utility in negotiation with the GRA. For example, the GRA might not concede until the client's deadline, pressuring a client to accept the minimum amount of resources. However, we assume that the GRA or a client can model its respective opponent as the one that applies a time-dependent concession-based strategy.

#### 3.2.2.3 Basic Negotiation Strategy

In this section, we describe the time-dependent strategy proposed by Faratin et al. [17] and adopted by many researchers [112, 141, 148, 149] with respect to our assumptions and terminology. The client uses this strategy to generate a proposal with some resource (see Formula (3.10)). Each round of negotiation $k$ the client proposes a certain resource $\hat{r}^c_{i,k}$[2] for task $i$. The client's proposed resource $\hat{r}^c_{i,k} > 0$ has to satisfy the corresponding negotiation interval $\left[R_i^{max}, R_i^{min}\right]$. In this interval, $R_i^{max}$ is an maximum (aspiration) resource which is the most desirable for task $i$ and $R_i^{min}$ is the minimum (reservation) resource that can be accepted by the client, but it is the most undesirable for task

[2]Here, the upper index means "client".

$i$. The first client's proposal is $R_i^{max}$ and then the client makes concessions towards the GRA until an agreement is reached or negotiation is terminated with a failure. Negotiation starts when the round of negotiation is $k = 0$ and continues with an elementary time step $\tau^{elem}$, which equals to one virtual second, until $t_{dl}^c$ is reached (if an agreement or termination has not been reached). When the client's negotiation deadline is reached $t_{dl}^c$, the client proposes its reservation amount of resource $R_i^{min}$. This time-dependent strategy allows the client to apply three different time-dependent tactics (see Formula (3.12)) and these tactics are determined by a coefficient $\beta_{i,k}^c$. The client's time-dependent negotiation strategy for resources is described in Equation (3.10).

$$\hat{r}_{i,k}^c = R_i^{max} + \left(\frac{\tau^{elem} * k}{t_{dl}^c}\right)^{\beta_{i,k}^c} \times \left(R_i^{min} - R_i^{max}\right), \tag{3.10}$$

The GRA also generates its proposals $\hat{r}_{i,k}^{gra} \geq 0$ using the time-dependent strategy, but compared to a client the GRA's reservation amount of resource is the maximum resource $G_{i,k}^{max} > 0$ and its aspiration resource is the minimum resource $G_i^{min} > 0$ that can be offered for task $i$. We assume that the GRA's minimum resource $G_i^{min}$ is always equal or larger than the client's minimum acceptable resource $R_i^{min}$, because the GRA has no reason to offer the resource that cannot be accepted by the client. The GRA's maximum resource $G_{i,k}^{max}$ is assumed to be equal or smaller than the client's maximum amount of resource $R_i^{max}$, because the GRA has no reason to offer the resource that is larger then the client's maximum one when Grid resources are scarce. The change in the GRA's reservation value $G_{i,k}^{max}$ and the direction of this change is generated randomly per negotiation round, if this change occurs. The value of $G_{i,k}^{max}$ can change of up to some percentage of the client's maximum value, and if it becomes smaller than $G_i^{min}$ (and $R_i^{min}$), a negotiation process is terminated with a failure due to the resource exhaustion. The GRA also has its negotiation deadline $t_{dl}^{gra} > 0$ which is considered to be the same as the client's deadline in this work, and its own coefficient $\beta_{i,k}^{gra}$ that determines its tactic.

$$\hat{r}_{i,k}^{gra} = G_i^{min} + \left(\frac{\tau^{elem} * k}{t_{dl}^{gra}}\right)^{\beta_{i,k}^{gra}} \times \left(G_{i,k}^{max} - G_i^{min}\right), \tag{3.11}$$

The coefficients $\beta_{i,k}^c$ and $\beta_{i,k}^{gra}$ define the level of greediness of the client or the GRA in respect to each other and, as a result, the amount of concession towards each other

every negotiation round [17].

$$\begin{aligned} &\beta_{i,k}^{c,gra} > 1, && \textit{Greedy tactic}, \\ &\beta_{i,k}^{c,gra} = 1, && \textit{Indifferent tactic}, \\ &0 < \beta_{i,k}^{c,gra} < 1, && \textit{Generous tactic}. \end{aligned} \tag{3.12}$$

The values of $\beta_{i,k}^{c}$ and $\beta_{i,k}^{gra}$ may change for the different tasks during negotiation, because they depend on the resource availability changes in the Grid. For example, if the amount of available resources decreases, the GRA becomes less generous but the client becomes more generous. This example can be explained that the client has a less chance to obtain a resource and it has to concede a lot to reach an agreement, while the GRA has to keep resources for other clients. The adaptive strategy for the client is developed to estimate the percentage change of $\beta_{i,k}^{c}$, considering the resource availability changes in a Grid. In this chapter, we describe how the GRA's level of greediness (see Section 3.3.1.1) and the change of its reservation amount of resource (see Section 3.3.1.2) are estimated by a client. We also discuss a fuzzy design which allows a client to adapt to the changes of resource availability based on the estimations above mentioned (see Section 3.3.2). We also demonstrate an algorithm which allows the client to vary dynamically the input and output of the fuzzy controller based on the predicted best outcomes considering the history of negotiation (see Section 3.3.3).

TABLE 3.1: List of notation for Section 3.2

| **Symbol** | **Notation** |
|---|---|
| $i$ | An identifier of the client's task in one job. |
| $Spec_i$ | A specification of task $i \in \mathbb{N}$ which defines task requirements to be run in a Grid. |
| $Task_i$ | A tuple of private attributes for task $i \in \mathbb{N}$. |
| $R_i^{min}$ | A minimum acceptable amount of resource (reservation value) required by a client to run task $i \in \mathbb{N}$, where $R_i^{min} > 0$. |
| $R_i^{max}$ | An maximum amount of resource (aspiration value) which is the most desirable for a client to run task $i \in \mathbb{N}$, where $R_i^{max} > 0$. |
| $Job$ | A specification of a set of tasks, which comprises the specifications of all client's tasks. |
| $Pr_{i,j}$ | A proposal of a specific resource amount for task $i$ which can be sent by a client $Pr_{i,j}^{c}$ or the GRA $Pr_{i,j}^{gra}$ in round $j$, where $i, j \in \mathbb{N}$. |

Continued on the next page

Continued from the previous page

| Symbol | Notation |
|---|---|
| $\hat{r}_{i,j}$ | An amount of resource proposed by a client $\hat{r}^c_{i,j} > 0$ or the GRA $\hat{r}^{gra}_{i,j} \geq 0$ in round $j$, where $\hat{r}_{i,j} \geq 0$, $i, j \in \mathbb{N}$. |
| $r_i$ | An amount of allocated resource for task $i \in \mathbb{N}$, where $r_i \geq 0$. |
| $N_{suc}$ | The number of successful tasks, which obtained resources during a single negotiation, where $N_{suc} \in \mathbb{N}$. |
| $U_{Total}$ | A client's utility function which maps the set of obtained resources for all tasks and the number of successful tasks to a specific value from the interval $[0, 1]$. |
| $U^r_i$ | An estimate which indicates the client's level of satisfaction in respect of the obtained resource amount for task $i \in \mathbb{N}$, where $U^r_i \in [0, 1]$. |
| $k^i_r$ | The value of estimate $U^r_i \in [0, 1]$, when task $i \in \mathbb{N}$ is allocated the minimum acceptable amount of resource $R^{min}_i > 0$, where $k^i_r \in ]0, 1]$. |
| $U_R$, $U_T$ | The variables which determine the quality of the obtained resources for all tasks and the level of success in negotiation respectively, where $U_R$, $U_T \in [0, 1]$. |
| $t^c_{dl}$, $t^{gra}_{dl}$ | The deadline of negotiation for a client and the GRA respectively, where $t^c_{dl} > 0$, $t^{gra}_{dl} > 0$. |
| $\beta^c_{i,k}$, $\beta^{gra}_{i,k}$ | The level of greediness of a client and the GRA for task $i \in \mathbb{N}$ in round $k \in \mathbb{N}$, which determines their levels of concession towards each other. |
| $G^{max}_{i,k}$ | The maximum resource (reservation value) that can be proposed by the GRA to a client for task $i \in \mathbb{N}$ at the time of round $k \in \mathbb{N}$, where $G^{max}_{i,k} > 0$. |
| $G^{min}_i$ | The minimum resource (aspiration value) that can be proposed by the GRA to a client for task $i \in \mathbb{N}$, where $G^{min}_i > 0$. |

## 3.3 Adaptive Negotiation Strategy

Our adaptive negotiation strategy for a client aims to estimate the GRA's negotiation parameters (see Section 3.3.1) based on its proposals and assumptions we have described in Section 3.2.2.2 in respect of the level of uncertainty between the negotiators in order to evaluate the change in resource availability. Then, a client has to make a decision how to change its own level of greediness in order to avoid the resource exhaustion

during negotiation and this decision-making mechanism is implemented as a fuzzy-based controller in Section 3.3.2. The risk of resource exhaustion is estimated by our evaluation function described in Section 3.3.3, which determines the best combination of fuzzy sets per negotiation round.

### 3.3.1 Estimating the GRA's Negotiation Parameters

This section discusses how a client estimates the GRA's negotiation parameters, i.e. the level of greediness and reservation amount of resource. The client has to estimate those parameters to figure out any changes in the resource availability because the client does not have a direct access to this information (see Section 3.2.2.2).

#### 3.3.1.1 Level of Greediness

A tactic of the GRA is determined by the level of greediness $\beta_{i,k}^{gra}$ which is described in Formula (3.12). We assume that a client does not know the GRA's level of greediness because this information is private. However, the client models the GRA as it uses a time-dependent strategy, because we assume that all negotiating parties use this strategy within a particular Grid. We believe that our assumption is realistic because negotiation for the time-dependent issues such as resources has to follow the time-dependent strategy. It is also consistent to assume that the negotiators should broadly expect some patterns of behaviour of their opponents in order to make any decisions. In this way, the client assumes that the GRA's proposed resource $\hat{r}_{i,k}^{gra}$ rises rapidly when it applies a generous tactic (i.e. the level of greediness is between 0 and 1) or slowly when it applies a greedy tactic (i.e. the level of greediness is higher than 1). If the GRA applies an indifferent tactic, then the GRA's proposed resource increases on the same amount each negotiation round. Consequently, the client can compare the GRA's proposals in the previous $k-1$ and current $k$ rounds of negotiation to estimate its level of greediness.

$$\begin{cases} \hat{r}_{i,k}^{gra} = G_i^{min} + \left(\frac{t*k}{t_{dl}^{gra}}\right)^{\beta_{i,k}^{gra}} \times \left(G_{i,k}^{max} - G_i^{min}\right), \\ \hat{r}_{i,k-1}^{gra} = G_i^{min} + \left(\frac{t*(k-1)}{t_{dl}^{gra}}\right)^{\beta_{i,k-1}^{gra}} \times \left(G_{i,k-1}^{max} - G_i^{min}\right). \end{cases} \tag{3.13}$$

Assume that $\beta_{i,k}^{gra} = \beta_{i,k-1}^{gra}$ and $G_{i,k}^{max} = G_{i,k-1}^{max}$, i.e. the resource availability does not change in the current round of negotiation compared to the previous one. In this way,

the system of equations (see Formula (3.13)) can be presented as in Formula (3.14).

$$\frac{\hat{r}_{i,k}^{gra} - G_i^{min}}{\hat{r}_{i,k-1}^{gra} - G_i^{min}} = \left(\frac{k}{k-1}\right)^{\beta_{i,k}^{gra}} . \tag{3.14}$$

Then, the level of greediness $\beta_{i,k}^{gra}$ can be derived from Formula (3.14).

$$\beta_{i,k}^{gra} = \ln\left(\frac{\hat{r}_{i,k}^{gra} - G_i^{min}}{\hat{r}_{i,k-1}^{gra} - G_i^{min}}\right) / \ln\left(\frac{k}{k-1}\right). \tag{3.15}$$

According to Formula (3.15), $\beta_{i,k}^{gra}$ can be estimated only if the resource availability does not change in the current negotiation round $k$ because it would mean that $\beta_{i,k}^{gra}$ and $G_{i,k}^{max}$ are changed by the GRA. Consequently, the client is not able to compare the previous and current GRA proposals.

#### 3.3.1.2 Reservation Value

We assume that a client does not know the GRA's reservation amount of resource $G_{i,k}^{max}$ because it is private information. Therefore, the client is not able to calculate exactly on how much the GRA's reservation amount of resource has changed compared to the previous negotiation round. However, the client is able to estimate the GRA's level of greediness $\beta_{i,k}^{gra}$ after the first two negotiation rounds and then each next round after the round when the resource availability changes. When the GRA's level of greediness is estimated, a client can predict the next proposal of the GRA using Formula (3.14). If the GRA offers resource which is not approximately the same as the predicted resource, the client decides that the GRA's level of greediness and reservation amount of resource changed due to the change of the resource availability in the Grid. For example, if the proposed resource is smaller than the predicted one, it denotes that the GRA's reservation amount of resource decreased. Any changes in the GRA's negotiation behaviour (i.e. the GRA's reservation value and level of greediness) must be motivated by some relevant factor, and the change in resource availability is an intuitive motivation for a non-commercial Grid.

Assume that the current round of negotiation is $k$ and the next round of negotiation is $k+1$ respectively. The GRA's level of greediness $\beta_{i,k}^{gra}$ is estimated in the current negotiation round $k$ and the client assumes that it will not change in the next round $k+1$, i.e. $\beta_{i,k+1}^{gra'} = \beta_{i,k}^{gra}$ (the upper index ($'$) denotes an expected value, not an actual

value in the round $k+1$). Considering Formula (3.14), the increment of the GRA's expected proposal compared to its first proposal $\Delta\hat{r}^{gra'}_{i,k+1} = \hat{r}^{gra'}_{i,k+1} - G^{min}_i$ is estimated as in Formula (3.16).

$$\Delta\hat{r}^{gra'}_{i,k+1} = \left(\frac{k+1}{k}\right)^{\beta^{gra'}_{i,k+1}} \times \Delta\hat{r}^{gra}_{i,k}, \tag{3.16}$$

where $\Delta\hat{r}^{gra}_{i,k} = \hat{r}^{gra}_{i,k} - G^{min}_i$.

The client knows that the GRA will propose its reservation amount of resource when its deadline of negotiation is reached because the GRA uses a time-dependent strategy. However, we assume that the client does not know the GRA's negotiation deadline $t^{gra}_{dl}$ and it substitutes this deadline with its own $t^{gra}_{dl} = t^{c}_{dl}$, i.e. our current work is limited to negotiation with equal deadlines. Therefore, the client makes a decision that the GRA will propose its reservation amount of resource in $t^{c}_{dl}$ negotiation round. Then the client can estimate the expected difference $\Delta\hat{r}^{max'}_{i,k+1}$ between the GRA's reservation $G^{max'}_{i,k+1}$ and aspiration $G^{min}_i$ resources in the round $k+1$ if the client assumes that the resource availability will not change in that round. Considering Formula (3.16),

$$\Delta\hat{r}^{max'}_{i,k+1} = \left(\frac{t^{c}_{dl}}{k+1}\right)^{\beta^{gra'}_{i,k+1}} \times \Delta\hat{r}^{gra'}_{i,k+1}. \tag{3.17}$$

Assume that the increment of the GRA's expected proposal $\Delta\hat{r}^{gra'}_{i,k+1}$ is not approximately the same as the increment of its actual proposal $\Delta\hat{r}^{gra}_{i,k+1}$ in the negotiation round $k+1$, i.e. the resource availability has changed. It means that $\hat{r}^{max}_{i,k+1}$ is not the same as it was expected $\hat{r}^{max'}_{i,k+1}$ by the client because the GRA's reservation amount of resource $G^{max}_{i,k+1}$ has changed. Moreover, the GRA's level of greediness $\beta^{gra}_{i,k+1}$ also changes compared to the expected level $\beta^{gra'}_{i,k+1}$ in the previous negotiation round. According to Formula (3.15), we can estimate $\beta^{gra}_{i,k+1}$ only in the next round after the resource availability changes, i.e. in the round $k+2$. Assume that the client reaches the negotiation round $k+2$ and calculates $\beta^{gra}_{i,k+1}$ and $\hat{r}^{gra'}_{i,k+1}$. Then the client can estimate the difference $\Delta\hat{r}^{max}_{i,k+1} = G^{max}_{i,k+1} - G^{min}_i$ that was in the round $k+1$ as presented in Formula (3.18).

$$\Delta\hat{r}^{max}_{i,k+1} = \left(\frac{t^{c}_{dl}}{k+1}\right)^{\beta^{gra}_{i,k+1}} \times \Delta\hat{r}^{gra}_{i,k+1}. \tag{3.18}$$

The client is able to estimate the proportion of the expected $\Delta\hat{r}^{max'}_{i,k+1}$ and actual $\Delta\hat{r}^{max}_{i,k+1}$ differences of the GRA's reservation and aspiration resources in the round $k+1$ (see

Formulae (3.17)-(3.18)).

$$\frac{\Delta\hat{r}^{max}_{i,k+1}}{\Delta\hat{r}^{max'}_{i,k+1}} = \left(\frac{t^c_{dl}}{k+1}\right)^{\beta^{gra}_{i,k+1}-\beta^{gra'}_{i,k+1}} \times \frac{\Delta\hat{r}^{gra}_{i,k+1}}{\Delta\hat{r}^{gra'}_{i,k+1}}. \tag{3.19}$$

We present the proportion in Formula (3.19) as a percentage $\delta_{i,k+1}$ compared to this proportion in the previous round $k$ when the resource availability did not change.

$$\delta_{i,k+1} = \left(\frac{\Delta\hat{r}^{max}_{i,k+1}}{\Delta\hat{r}^{max'}_{i,k+1}} - 1\right) \times 100\%, \tag{3.20}$$

where $\delta_{i,k+1} > 0$ if the GRA's reservation amount of resource increases and $\delta_{i,k+1} < 0$ if the GRA's reservation amount of resource decreases.

### 3.3.2 Designing Fuzzy Model

The client's level of greediness $\beta^c_{i,k}$ can be increased or decreased depending on the resource availability in the Grid. According to the intuition of Narayanan and Jennings [112], if the resource availability decreases, the client has to become more generous to reach an agreement with the GRA before the resources are exhausted. If the resource availability increases, the client has to become less generous to avoid losing its utility without a risk of resource exhaustion. However, the issue is how to distinguish between the "large" or "small" increase/decrease of the resource availability in the Grid. The client estimates only the relative change $\delta_{i,k}$ (see Formula (3.20)) of the GRA's negotiation interval $\left[G^{min}_i, G^{max}_{i,k}\right]$. In other words, the client is not absolutely certain whether a specific value of $\delta_{i,k}$ denotes a significant or an insignificant change of the GRA's reservation amount of resource. Therefore, the client evaluates whether the resource availability has changed substantially using a fuzzy representation of $\delta_{i,k}$. That is, the client is only able to state with some degree of certainty that the GRA's reservation amount of resource has decreased or increased significantly. If the change of resources is insignificant than we believe that the client will not benefit from a significant change of its level of greediness because the client is uncertain whether a risk of resource exhaustion is large or small.

The client also has to take into account its current level of greediness to decide how this level can be changed in the current negotiation round. For example, if the client applies generous tactic and the resource availability decreases, the client may not need

to become more generous because the client may lose its utility. However, if the client applies greedy tactic and the resource availability decreases, the client has to become more generous because the client may fail negotiation. We believe that the client may significantly change its level of greediness if it applies a greedy or generous tactic but a significant change will not be beneficial if the client applies an indifferent tactic. In this way, we try to find a balance between a risk of negotiation failure and loss of utility. Therefore, the client has to decide with some degree of certainty whether its current tactic is "more" or "less" generous or greedy to change more or less significantly its level of greediness.

The output of our fuzzy mechanism shows how the client's current level of greediness has to be changed to cope with availability of Grid resources. The output is also represented as "fuzzy" because of the fuzzy input (i.e. the resource availability and the client's current level of greediness). Moreover, Grids can have different characteristics that may influence on the result in the different way and those characteristics may be unknown to the client to take them into account. In other words, the lack of knowledge about a Grid environment leads the client to use *fuzzy logic* [177, 178] to estimate the change of its level of greediness.

A fuzzy logic operates with *fuzzy sets* which allow their members to have a partial membership. For example, the value of $\beta^c_{i,k} = 1.5$ can be considered as the member of the set of greedy tactics with the 0.5 degree of membership and as the member of the set of indifferent tactics with the same degree of membership. In other words, $\beta^c_{i,k} = 1.5$ does not belong only to the one certain set. In fuzzy logic, the membership function defines the level of our confidence that a certain value belongs to a particular fuzzy set. We consider the change in the increment of the GRA's reservation value $\delta_{i,k+1}$ and the current client's level of greediness $\beta^c_{i,k}$ as an input to a fuzzy mechanism (see Section 3.3.2.1). In this way, the change of $\beta^c_{i,k}$ is considered as an output of a fuzzy mechanism (see Section 3.3.2.2). This fuzzy mechanism comprises three blocks, i.e. *fuzzification*, *inference* and *defuzzification*. In the first block, the input parameters are fuzzified, i.e. they converted from the crisp values to the degree of membership in the corresponding fuzzy sets. In the next block, fuzzy control rules map an input to the corresponding output (see Sections 3.3.2.3, 3.3.2.4), i.e. they define a response of the client to any input changes. The last block allows the client to obtain a crisp value of the change of the client's level of greediness (see Section 3.3.2.5). Finally, the result is a certain percentage $\eta_{i,k}\%$ on which the value of $\beta^c_{i,k}$ has to be increased or decreased in the current negotiation round.

#### 3.3.2.1 Input Membership Functions

Assume that $X$ is a non-fuzzy set of objects in which $x$ is a generic element of $X$, i.e. $x \in X$ [177, 178].

*Definition* 3.4. A *fuzzy set* $A$ is the subset of $X$ which is defined ('characterised') by a *membership function* $\mu_A(x)$. This function maps each element $x$ of $X$ to the interval of real numbers from 0 to 1 and denotes the degree of membership of $x$ in $A$. That is, $\mu_A(x) : \ x \in X \ \rightarrow \ [0, 1]$.

**The Change of the GRA's Reservation Value** The change of the GRA's reservation amount of resource denotes that resources become more or less available in the Grid. According to Formula (3.20), we estimate a relative change of the GRA's negotiation interval, i.e. $G_{i,k}^{max} - G_i^{min}$. However, we refer to this estimation as the change of the GRA's reservation amount of resource because only its reservation amount of resource changes within this interval. We design three fuzzy sets to describe these changes: "decrease" (D), "zero" (Z) and "increase" (I), where the intersections denote an uncertainty of the client which are depicted in Figure 3.3. This figure shows an example of the input membership function for the estimate $\delta_{i,k}$ of the change in the GRA's reservation amount of resource, where *OX-axis* denotes a value of $\delta_{i,k}$ and *OY-axis* denotes its degree of membership $\mu(\delta_{i,k})$ in the above mentioned fuzzy sets. The degree of membership which is equal to one means that the client is absolutely convinced that the value of $\delta_{i,k}$ belongs to a particular fuzzy set. Consequently, the degree of membership which is equal to zero means that the client is absolutely convinced that the value of $\delta_{i,k}$ does not belong to a particular fuzzy set.

As an example, we assume that the change of the reservation amount of resource which is less than 50% might be considered as a slight change ("zero") and more than 50% as a significant change ("decrease" or "increase"). However, those intervals can be different for the different circumstances e.g., a decreasing or increasing tendency in the resource availability changes. We also assume that the maximum decrease of $\delta_{i,k}$ is 100% during a one step of negotiation, i.e. the reservation amount of resource cannot become a negative number. However, the increase of $\delta_{i,k}$ can be larger than 100% because a Grid is considered to be unbounded in respect to the amount of resources that can join it. According to Figure 3.3, the input membership functions for "decrease" $\mu_{dec}(\delta_{i,k})$,

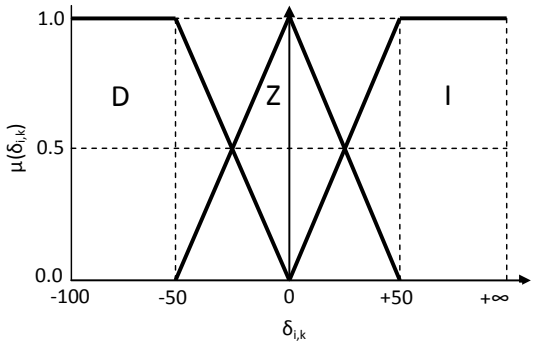



FIGURE 3.3: Input membership function for $\delta_{i,k}$, %

"zero" $\mu_{zero}(\delta_{i,k})$ and "increase" $\mu_{inc}(\delta_{i,k})$ fuzzy sets are described in Formula (3.21).

$$\mu_{dec}(\delta_{i,k}) = \begin{cases} 1, & -100 \leq \delta_{i,k} \leq -50; \\ -0.02 \times \delta_{i,k}, & -50 < \delta_{i,k} < 0; \\ 0, & \delta_{i,k} \geq 0. \end{cases}$$

$$\mu_{zero}(\delta_{i,k}) = \begin{cases} 0, & \delta_{i,k} \leq -50 \; or \; \delta_{i,k} \geq +50; \\ 0.02 \times \delta_{i,k} + 1, & -50 < \delta_{i,k} < 0; \\ -0.02 \times \delta_{i,k} + 1, & 0 \leq \delta_{i,k} < +50. \end{cases} \tag{3.21}$$

$$\mu_{inc}(\delta_{i,k}) = \begin{cases} 0, & \delta_{i,k} \leq 0; \\ 0.02 \times \delta_{i,k}, & 0 < \delta_{i,k} < +50; \\ 1, & \delta_{i,k} \geq +50. \end{cases}$$

where $\mu_{dec}(\delta_{i,k}) \in \mathbb{R}$, $\mu_{zero}(\delta_{i,k}) \in \mathbb{R}$, $\mu_{inc}(\delta_{i,k}) \in \mathbb{R} \; \rightarrow \; [0, 1]$.

**The Client's Level of Greediness** The client's level of greediness $\beta^c_{i,k}$ affects its amount of concession over time. We design three fuzzy sets based on the three types of tactic $\beta^c_{i,k}$, i.e. "generous" (GEN), "indifferent" (IND) and "greedy" (GR) which are depicted in Figure 3.4. This figure demonstrates an example of the membership function of the client's level of greediness $\beta^c_{i,k}$, where *OX-axis* denotes the value of $\beta^c_{i,k}$ and *OY-axis* denotes its degree of membership $\mu(\beta^c_{i,k})$.

Although generous, indifferent and greedy tactics are defined explicitly in other work [17, 148], the shapes of the curves for generous tactics with e.g., $0.8 \leq \beta^c_{i,k} < 1.0$, are approximately close to an indifferent (linear) tactic. In other words, these tactics can be considered more as indifferent than as generous and otherwise. In the same way, greedy

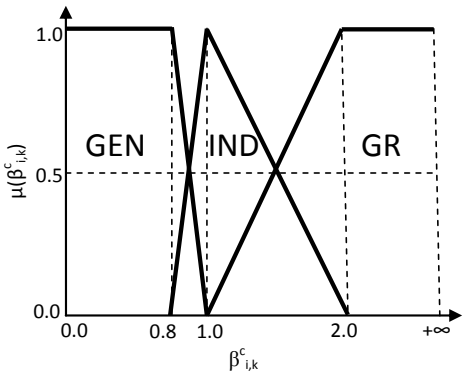



FIGURE 3.4: Input membership function for $\beta^c_{i,k}$

tactics with e.g., $1.0 < \beta^c_{i,k} < 2.0$, can be considered as indifferent tactics. For example, if the client's level of greediness is equal to 0.9, the client is uncertain whether it has to change its level significantly in the case when the resource availability increases. The client's judgment whether a particular value of $\beta^c_{i,k}$ should be considered as more or less generous tactic may vary depending on the circumstances in which a client has to take more strict actions (e.g. a rapid decrease or increase of $\beta^c_{i,k}$) or more moderate actions (e.g. a medium decrease or increase of $\beta^c_{i,k}$). For example, if the resource availability decreases, $\beta^c_{i,k} = 3.0$ can be considered as a high level of greediness (i.e. "greedy"), but if the resource availability increases, then this level of greediness can be considered as not so high (i.e. "indifferent").

According to Figure 3.4, the input membership functions for "generous" $\mu_{gen}(\beta^c_{i,k})$, "indifferent" $\mu_{ind}(\beta^c_{i,k})$ and "greedy" $\mu_{gr}(\beta^c_{i,k})$ tactics are described in Formula (3.22).

$$
\begin{aligned}
\mu_{gen}\left(\beta^c_{i,k}\right) &= \begin{cases} 1, & 0 < \beta^c_{i,k} < 0.8; \\ -5 \times \beta^c_{i,k} + 5, & 0.8 < \beta^c_{i,k} \leq 1.0; \\ 0, & \beta^c_{i,k} \geq 1.0. \end{cases} \\
\mu_{ind}\left(\beta^c_{i,k}\right) &= \begin{cases} 5 \times \beta^c_{i,k} - 4, & 0.8 < \beta^c_{i,k} \leq 1.0; \\ -\beta^c_{i,k} + 2, & 1.0 < \beta^c_{i,k} < 2.0; \\ 0, & \beta^c_{i,k} \leq 0.8 \text{ or } \beta^c_{i,k} \geq 2.0. \end{cases} \\
\mu_{gr}\left(\beta^c_{i,k}\right) &= \begin{cases} \beta^c_{i,k} - 1, & 1.0 < \beta^c_{i,k} \leq 2.0; \\ 1, & \beta^c_{i,k} > 2.0; \\ 0, & \beta^c_{i,k} \leq 1.0. \end{cases}
\end{aligned}
\tag{3.22}
$$

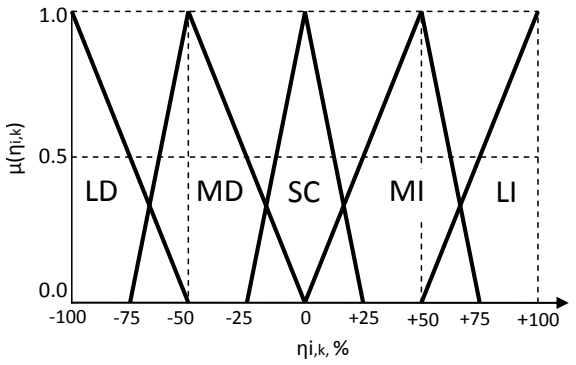



FIGURE 3.5: Output membership function for $\eta_{i,k}$, %

where $\mu_{gen}\left(\beta^c_{i,k}\right) \in \mathbb{R}$, $\mu_{ind}\left(\beta^c_{i,k}\right) \in \mathbb{R}$, $\mu_{gr}\left(\beta^c_{i,k}\right) \in \mathbb{R} \rightarrow\ [0,1]$.

#### 3.3.2.2 Output Membership Function

The output membership function allows a client to calculate a new level of greediness $\beta^{c'}_{i,k}$ as the percentage $\eta_{i,k}\%$ of $\beta^c_{i,k}$ on which the current level of greediness $\beta^c_{i,k}$ has to be increased or decreased (see Formula (3.23)).

$$\beta^{c'}_{i,k} = \beta^c_{i,k} \times \left(1 + \frac{\eta_{i,k}\%}{100\%}\right), \tag{3.23}$$

In Formula (3.23), a positive $\eta_{i,k}\%$ denotes an increase of $\beta^c_{i,k}$ and a negative $\eta_{i,k}\%$ denotes a decrease of $\beta^c_{i,k}$ (see Figure 3.5). The maximum allowed decrease of $\beta^c_{i,k}$ is equal to -100% of its current value, i.e. the level of greediness $\beta^c_{i,k}$ cannot be negative according to the possible tactics listed in Formula (3.12). However, the level of greediness $\beta^c_{i,k}$ can be potentially increased on more than +100%, but we do not consider this case because we believe that such rapid increase of $\beta^c_{i,k}$ would rise a risk of a failure of negotiation, because the client's tactic would be "greedy" (see Figure 3.4) for the most time of the negotiation process. Consequently, the longer this process continues, the higher can be a risk of resource exhaustion.

We design five fuzzy sets to calculate $\eta_{i,k}\%$, i.e. "large decrease" (LD), "medium decrease" (MD), "small change" (SC), "medium increase" (MI) and "large increase" (LI) (see Figure 3.5). Considering the number of input fuzzy sets (3 fuzzy sets for $\delta_{i,k}$ and 3 fuzzy sets for $\beta^c_{i,k}$), the possible number of their combinations is equal to 9. Therefore,

the maximum number of output fuzzy sets is equal to 9. However, we assume that if the resource availability does not change significantly, then the client cannot be certain whether a risk of resource exhaustion is large enough to increase or decrease its level of greediness significantly. In this way, we assume that each input combination that includes "zero" fuzzy set for $\delta_{i,k}$ (see Figure 3.3) should have "small change" of $\beta^c_{i,k}$ as its output. In this way, the client needs only 7 output fuzzy sets. Moreover, an "indifferent" fuzzy set for $\beta^c_{i,k}$ (see Figure 3.4) refers to the client's uncertainty whether it has to increase or decrease its level of greediness significantly. That is, this input fuzzy set refers to all "medium" outputs, while "generous" and "greedy" input fuzzy sets may lead to "large increase" or "large decrease" outputs. Thus, we believe that the client does not need more than the 5 output fuzzy sets to respond appropriately to any input combination.

For example, Figure 3.5 shows that the value of $\beta^c_{i,k}$ can be increased or decreased on more than a half of itself that can be considered as a large or medium change of $\beta^c_{i,k}$. In the same way, the value of $\beta^c_{i,k}$ can be increased or decreased on less than a half of itself that can be considered as a medium or small change of $\beta^c_{i,k}$. These intervals of uncertainty (e.g. $[-25, 25]$ for SC) are taken as an example and an algorithm of their estimation is described in Section 3.3.3. An example of the output membership functions $\mu_{ld}(\eta_{i,k})$, $\mu_{md}(\eta_{i,k})$, $\mu_{sc}(\eta_{i,k})$, $\mu_{mi}(\eta_{i,k})$ and $\mu_{li}(\eta_{i,k})$ for the corresponding fuzzy sets "large decrease", "medium decrease", "small change", "medium increase" and "large increase" is described in Formula (3.24).

$$
\begin{aligned}
\mu_{ld}(\eta_{i,k}) &= \begin{cases} -0.02 \times \eta_{i,k} - 1, & -100 \le \eta_{i,k} < -50; \\ 0, & \eta_{i,k} \ge -50. \end{cases} \\
\mu_{md}(\eta_{i,k}) &= \begin{cases} 0.04 \times \eta_{i,k} + 3, & -75 < \eta_{i,k} \le -50; \\ -0.02 \times \eta_{i,k}, & -50 < \eta_{i,k} < 0; \\ 0, & \eta_{i,k} \ge 0 \ or \ \eta_{i,k} \le -75. \end{cases} \\
\mu_{sc}(\eta_{i,k}) &= \begin{cases} 0.04 \times \eta_{i,k} + 1, & -25 < \eta_{i,k} \le 0; \\ -0.04 \times \eta_{i,k} + 1, & 0 < \eta_{i,k} < +25; \\ 0, & \eta_{i,k} \ge +25 \ or \ \eta_{i,k} \le -25. \end{cases} \\
\mu_{mi}(\eta_{i,k}) &= \begin{cases} 0.02 \times \eta_{i,k}, & 0 < \eta_{i,k} \le +50; \\ -0.04 \times \eta_{i,k} + 3, & +50 < \eta_{i,k} < +75; \\ 0, & \eta_{i,k} \le 0 \ or \ \eta_{i,k} \ge +75. \end{cases} \\
\mu_{li}(\eta_{i,k}) &= \begin{cases} 0.02 \times \eta_{i,k} - 1, & +50 < \eta_{i,k} \le +100; \\ 0, & \eta_{i,k} \le +50. \end{cases}
\end{aligned}
\tag{3.24}
$$

TABLE 3.2: Fuzzy control rules

| | D | Z | I |
|---|---|---|---|
| GEN | MD | SC | LI |
| IND | MD | SC | MI |
| GR | LD | SC | MI |

#### 3.3.2.3 Fuzzy Control Rules

In fuzzy logic, the logical operations (e.g. 'conjunction', 'disjunction', 'negation') are performed in a distinct way compared to conventional logic. For example, 1 'AND' 1 means 'TRUE' in conventional logic. However, conventional logic cannot solve the case when 0.1 'AND' 0.8. In this case, fuzzy logic proposes its own inference for conventional logical operations [173, 177], i.e.

$$\begin{aligned} &\mu_{X'}(x)\ `AND'\ \mu_{Y'}(y) \ \rightarrow\ minimum\left(\mu_{X'}(x), \mu_{Y'}(y)\right); \\ &\mu_{X'}(x)\ `OR'\ \mu_{Y'}(y) \ \rightarrow\ maximum\left(\mu_{X'}(x), \mu_{Y'}(y)\right); \\ &`NOT'\ \mu_{X'}(x) \ \rightarrow\ 1-\mu_{X'}(x), \end{aligned} \tag{3.25}$$

where $x \in X$, $y \in Y$ and $X$, $Y$ are the non-fuzzy sets of objects and $\mu_{X'}(x)$, $\mu_{Y'}(y)$ are the degrees of membership for $x$ and $y$ in fuzzy sets $X^{'}$ and $Y^{'}$ respectively. In our work, fuzzy control rules have a structure $IF-THEN$ where $IF$ indicates the conditions under which a consequence $THEN$ occurs. These conditions are the fuzzified values of $\delta_{i,k}$ and $\beta^{c}_{i,k}$, i.e. the degrees of membership $\mu_{X'}(\delta_{i,k})$ and $\mu_{Y'}\left(\beta^{c}_{i,k}\right)$, which are connected by the logical operator $AND$ (see Formula (3.25)). A consequence to these conditions denotes how much the value of $\beta^{c}_{i,k}$ has to be changed (e.g. it has to be significantly increased ("large increase"(LI))). We distinguish 9 fuzzy control rules which are based on the two input membership functions for $\delta_{i,k}$ and $\beta^{c}_{i,k}$ (see Table 3.2). In Table 3.2, a fuzzy control rule can be presented as $IF\ (I)\ AND\ (GEN)\ THEN\ (LI)$. This control rule means that if the value of $\delta_{i,k}$ has increased (I) and the client applies a generous tactic $\beta^{c}_{i,k}$ (GEN), then the value of $\beta^{c}_{i,k}$ has to be significantly increased (i.e. "large increase"). This means that a client is losing significantly in its utility for being generous, while there is no risk of resource exhaustion ($\delta_{i,k}$ increases) and, therefore, a client has to become more greedy by increasing $\beta^{c}_{i,k}$. The first row of this table denotes the relative change of the GRA's reservation amount of resource, i.e. decrease (D), zero (Z) and increase (I) (see Formula (3.21)). The first column denotes the client's level of greediness, i.e. generous (GEN), indifferent (IND) and greedy (GR) (see Formula

(3.22)). Finally, the intersections in this table denote the change of the client's tactic $\eta_{i,k}$, i.e. large increase (LI), medium increase (MI), small change (SC), medium decrease (MD) and large decrease (LD) (see Formula (3.24)).

These fuzzy control rules are based on the intuition outlined by Narayanan and Jennings [112]. If the resource availability decreases (i.e. $\delta_{i,k}$ decreases (D)), then the client has to become more generous (i.e. $\beta^c_{i,k}$ has to be decreased) to avoid resource exhaustion. If the resource availability increases (i.e. $\delta_{i,k}$ increases (I)), then the client has to become less generous (i.e. $\beta^c_{i,k}$ has to be increased) to avoid losing its utility without a significant risk of resource exhaustion. In our work, the degree of decrease or increase of the client's greediness depends on its current level of greediness. For example, if the client is generous (GEN) and the resource availability decreases (D), then the client do not need to decrease its greediness significantly (i.e. "medium decrease" (MD)), which is different from the case when the client is currently greedy (GR). We believe that when the resource availability fluctuates insignificantly (Z), then the client also intends to change its greediness insignificantly ("small change" (SC)).

**Example of Fuzzy Control Rules' Application**

Assume that

- $\delta_{i,k} = -25$ then $\mu_{inc}(\delta_{i,k}) = 0.0$, $\mu_{zero}(\delta_{i,k}) = 0.5$ and $\mu_{dec}(\delta_{i,k}) = 0.5$ (see Formula (3.21));
- $\beta^c_{i,k} = 1.75$ then $\mu_{gen}\left(\beta^c_{i,k}\right) = 0.0$, $\mu_{ind}\left(\beta^c_{i,k}\right) = 0.25$ and $\mu_{gr}\left(\beta^c_{i,k}\right) = 0.75$ (see Formula (3.22)).

We apply fuzzy control rules, which are described in Table 3.2, to generate the inference for the input membership functions. The control rules with the non-zero results are described below, where $\mu^F_l\left(\delta_{i,k}, \beta^c_{i,k}\right)$ is the resulting input membership function, referring to the corresponding output fuzzy set $F == LI|MI|SC|MD|LD$ for each control rule $l = 0, 1, , ..., CR$ with the total number of rules equal to $CR$. This function determines a minimum value between the corresponding $\mu_{X'}(\delta_{i,k})$ and $\mu_{Y'}\left(\beta^c_{i,k}\right)$. The complete solution for this example is described in the following sections.

$if\ \ (\delta_{i,k} \in D) \wedge (\beta^c_{i,k} \in GR) \rightarrow\ \mu^{ld}_1\left(\delta_{i,k}, \beta^c_{i,k}\right) = 0.5;$

$if\ \ (\delta_{i,k} \in D) \wedge (\beta^c_{i,k} \in IND) \rightarrow\ \mu^{md}_2\left(\delta_{i,k}, \beta^c_{i,k}\right) = 0.25;$

$if\ \ (\delta_{i,k} \in Z)\ \wedge(\beta^c_{i,k} \in GR) \rightarrow\ \mu^{sc}_3\left(\delta_{i,k}, \beta^c_{i,k}\right) = 0.5;$

$if\ \ (\delta_{i,k} \in Z)\ \wedge(\beta^c_{i,k} \in IND) \rightarrow\ \mu^{sc}_4\left(\delta_{i,k}, \beta^c_{i,k}\right) = 0.25;$

#### 3.3.2.4 Inference

The Mamdani [177] inference method is intuitively understandable and computationally effective for the small number of control rules. However, if the number of input parameters rises, the number of fuzzy control rules increases exponentially which means that this method is not suitable for the large number of input parameters. In our implementation, there are only two input parameters, i.e. $\delta_{i,k}$ and $\beta^c_{i,k}$. Moreover, the Mamdani's inference method does not require complex calculations which may consume much time for a client to generate its proposal and increase the risk of losing its utility or failing a negotiation with the GRA. A discussion of this method and other fuzzy logic models are presented in Section 2.5.2. This inference method consists of two stages: implication and aggregation. The *implication* is applied to each particular fuzzy control rule using a *minimum* operator and *aggregation* combines inferences for all fuzzy control rules in the one resulting output membership function using a *maximum* operator.

**Implication** Here, the implication for each fuzzy control rule is performed with a function $\phi(\mu_F(\eta_{i,k}), \mu^F_l\left(\delta_{i,k}, \beta^c_{i,k}\right))$ for a control rule $l \in \mathbb{N}$, $l = 0, 1, ..., CR$ and output fuzzy set $F = LI|MI|SC|MD|LD$ of $\eta_{i,k}$ (see Section 3.3.2.2). The function $\phi(\cdot)$ applies the minimum operator to the resulting input membership function $\mu^F_l\left(\delta_{i,k}, \beta^c_{i,k}\right)$ (see Section 3.3.2.3) and to the corresponding output membership function $\mu_F(\eta_{i,k})$ for a fuzzy set $F$, according to a fuzzy control rule $l$. This minimum operator produces the truncated output membership function $\mu^l_F(\eta_{i,k})$ for an output fuzzy set $F$. Consequently, the stage of implication is described as in Formula (3.26).

$$\phi(\mu_F(\eta_{i,k}), \mu^F_l\left(\delta_{i,k}, \beta^c_{i,k}\right)) = min\left[\mu_F(\eta_{i,k}), \mu^F_l\left(\delta_{i,k}, \beta^c_{i,k}\right)\right]. \tag{3.26}$$

In other words, the implication function truncates the output membership function of $\eta_{i,k}$ decreasing the maximum degree of membership of $\eta_{i,k}$ in a particular output fuzzy set. Considering an example in the previous section, the implication for the first control rule[3], i.e. $if\ \ (\delta_{i,k} \in D) \wedge \left(\beta^c_{i,k} \in GR\right)\ \rightarrow\ (\eta_{i,k} \in LD)$, produces the truncated output

[3]The rules are numbered in the order which they are listed in previous Section.

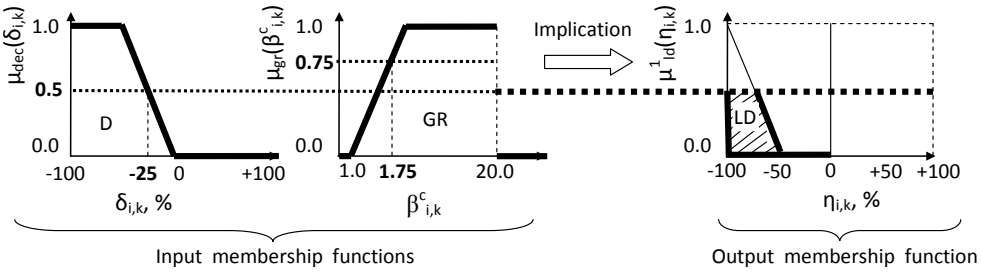



FIGURE 3.6: Implication process

membership function $\mu_{ld}^{1}(\eta_{i,k})$ of $\eta_{i,k}$ (see Figure 3.6). Thus, the only area under the value of $\mu_{ld}^{1}(\eta_{i,k}) = 0.5$ (i.e. the filled area) is considered for aggregation.

Figure 3.6 presents two input membership functions for $\delta_{i,k}$ and $\beta_{i,k}^{c}$ and the corresponding truncated output membership function $\mu_{ld}^{1}(\eta_{i,k})$ for the first control rule respectively. The logical operator 'AND' (i.e. minimum operator) is applied to $\mu_{dec}(\delta_{i,k}) = 0.5$ and $\mu_{gr}\left(\beta_{i,k}^{c}\right) = 0.75$, and as a result, $\mu_{1}^{ld}\left(\delta_{i,k}, \beta_{i,k}^{c}\right) = 0.5$. The next step is implication which produces $\mu_{ld}^{1}(\eta_{i,k}) = min\left[\mu_{ld}(\eta_{i,k}), 0.5\right]$, $\eta_{i,k} = [-100, +100]$, $\eta_{i,k} \in \mathbb{N}$. The result of implication is presented in Figure 3.6 as a truncated output membership function of $\eta_{i,k}$ for the LD fuzzy set.

**Aggregation** The next stage of Mamdani's inference method is the aggregation of inferences for all fuzzy control rules which is performed with a function $\varphi(\cdot)$. This function applies the maximum operator to all truncated output membership functions of $\eta_{i,k}$ obtained after implication. In this way, $\varphi(\cdot)$ aggregates all results in a single truncated output membership function of $\eta_{i,k}$, i.e. $\mu_{res}(\eta_{i,k})$. Consequently, the stage of aggregation is described in Formula (3.27).

$$\varphi(\left[\mu_{F_j}^{l}(\eta_{i,k})\right]_{CR,NS}) = max\left[\mu_{F_j}^{l}(\eta_{i,k})\right]_{CR,NS} \tag{3.27}$$

where $j$ denotes the identifier of the fuzzy output set (e.g. $F_1$ might be the "large increase" fuzzy set) and $NS$ is the total number of fuzzy output sets for a client. For example, if it is necessary to aggregate inferences for the first and second fuzzy control rules which are described in the previous section, i.e.

$(\delta_{i,k} \in D) \wedge (\beta_{i,k}^{c} \in GR) \rightarrow (\eta_{i,k} \in LD)$;

$(\delta_{i,k} \in D) \wedge (\beta_{i,k}^{c} \in IND) \rightarrow (\eta_{i,k} \in MD)$,

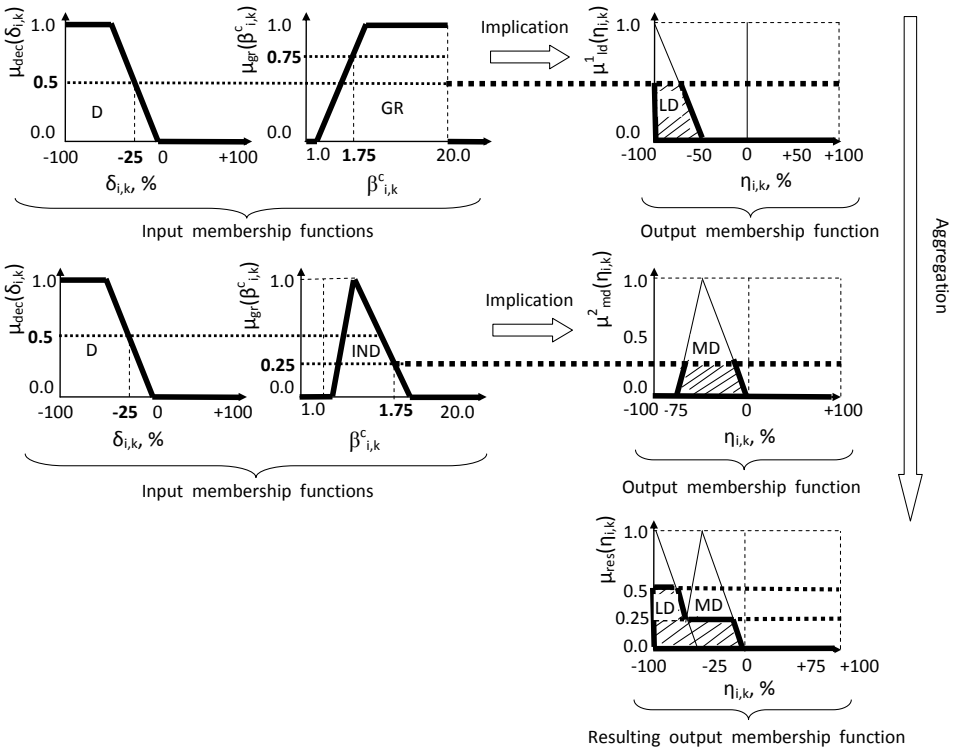



FIGURE 3.7: Aggregation process

the operator maximum is applied to $\mu^1_{ld}(\eta_{i,k})$ and $\mu^2_{md}(\eta_{i,k})$. The resulting output membership function $\mu_{res}(\eta_{i,k})$ is presented in Figure 3.7. The intuition of applying operator maximum is based on the fact that the inferences of all control rules have to be considered when the crisp value of $\eta_{i,k}$ is calculated. That is, the filled areas of both LD and MD fuzzy sets have to be considered in the process of derivation of the crisp value of $\eta_{i,k}$ (i.e. defuzzification), which is discussed in the next section.

#### 3.3.2.5 Defuzzification

A *defuzzification* is a process of derivation of the crisp value from the fuzzy result. The conventional defuzzification method of calculating the crisp value is the centre of gravity of the resulting output membership function (i.e. the filled output function). The centre of gravity method (COG) has its extended methods of defuzzification which are discussed and compared in Runkler and Glesner [203, 204]. According to these authors, the extended versions of the conventional method are more computationally effective.

However, the conventional COG method is proved to produce more plausible results. Moreover, this method is compatible with the Mamdani's type of fuzzy controllers. Therefore, this method is chosen for our current strategy. Applying the COG method, a value of $\eta_{i,k}$ is calculated with a defuzzification function $COG\left(\mu_{res}\left(\eta_{i,k}\right)\right)$ as in Formula (3.28).

$$COG\left(\mu_{res}\left(\eta_{i,k}\right)\right) = \frac{\int_{inf\left(\eta_{i,k}\right)}^{sup\left(\eta_{i,k}\right)} \mu_{res}\left(\eta_{i,k}\right)\eta_{i,k}d\eta_{i,k}}{\int_{inf\left(\eta_{i,k}\right)}^{sup\left(\eta_{i,k}\right)} \mu_{res}\left(\eta_{i,k}\right)d\eta_{i,k}}, \tag{3.28}$$

where $sup\left(\eta_{i,k}\right)$ is a supremum of $\eta_{i,k}$ and $inf\left(\eta_{i,k}\right)$ is an infimum of $\eta_{i,k}$. Considering the example discussed in the previous sections and Formula (3.28), the crisp value of $\eta_{i,k}$ is equal to -60.83% (see Formula (3.29)). That is, the value of $\beta^{c}_{i,k}$ has to be decreased on -60.83% in comparison with its previous value $\beta^{c}_{i,k-1}$.

$$\begin{aligned}\eta_{i,k} =& \frac{\int_{-100}^{-75} 0.5\eta_{i,k}d\eta_{i,k} + \int_{-75.0}^{-62.5}\left(-0.02\eta_{i,k}-1\right)\eta_{i,k}d\eta_{i,k} + \int_{-62.5}^{-12.5} 0.25\eta_{i,k}d\eta_{i,k}+}{\int_{-100}^{-75} 0.5d\eta_{i,k} + \int_{-75.0}^{-62.5}\left(-0.02\eta_{i,k}-1\right)d\eta_{i,k} + \int_{-62.5}^{-12.5} 0.25d\eta_{i,k}+} \\ & \frac{+\int_{-12.5}^{0}\left(-0.02\eta_{i,k}\right)\eta_{i,k}d\eta_{i,k}}{+\int_{-12.5}^{0}\left(-0.02\eta_{i,k}\right)d\eta_{i,k}} = -60.83\%.\end{aligned} \tag{3.29}$$

In Figure 3.8, the defuzzified value of $\eta_{i,k}$ is indicated with the bold vertical line for the resulting output membership function $\mu_{res}\left(\eta_{i,k}\right)$.

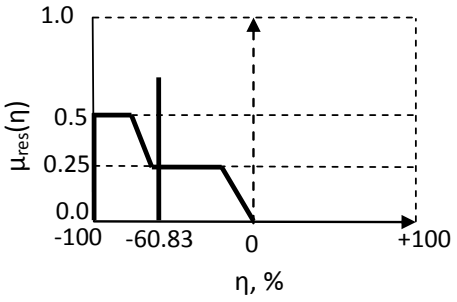



FIGURE 3.8: The defuzzified value of $\eta_{i,k}$, %

### 3.3.3 Automating Fuzzy Inference

The adaptive negotiation strategy, described in the previous sections, responds to the change of resource availability in the current negotiation round. However, the change of resource availability in the current negotiation round can be occasional rather than

following some tendency. Therefore, a client's response to this occasional change might not be adequate in terms of the tendency and could potentially lead to negotiation failure due to the resource exhaustion. Hence, a fuzzy model described in Section 3.3.2 needs to be re-adjusted in terms of its fuzzy sets every negotiation round, depending on the trends in resource availability changes. This enables a client to use a proper tactic in order to avoid negotiation failure and/or obtain the larger amounts of resource. A re-adjustment of the fuzzy sets means the change of their borders, i.e. their *uncertainty intervals*.

*Definition* 3.5. An uncertainty interval denotes the range of values $x \in [a, b]$ which belongs to a particular fuzzy set $A$, where $a$ and $b$ denote the borders of this fuzzy set, with some level of certainty $\mu_A(x)$.

For example, $\eta_{i,k} \in [0, +75]$ is the uncertainty interval of MI fuzzy set depicted in Figure 3.5. A *combination* of uncertainty intervals refers to the intervals of multiple fuzzy sets. Our adaptive negotiation strategy for a client predicts a possible outcome of negotiation (i.e. an amount of resource), if the client had used a specific tactic. Then, this strategy evaluates whether this outcome is the best one (e.g. the largest amount of resource) compared to others, generated by the different combinations of the *uncertainty intervals* of the membership functions, considering the estimated overall direction and average speed of resource dynamism over previous negotiation rounds.

#### 3.3.3.1 Characteristics of Resource Dynamism

Resource dynamism can be characterised by its *direction* (i.e. positive or negative) and *speed* (i.e. the change of resource availability per negotiation round). A client may estimate the change of resource availability based on the change of the GRA's reservation amount of resource $G_{i,k}^{max}$ (see Formula 3.20). This estimation compares the expected difference $\Delta\hat{r}_{i,k}^{max'}$ to the actual one $\Delta\hat{r}_{i,k}^{max}$ where the expected difference $\Delta\hat{r}_{i,k}^{max'}$ may change over time $k$. This estimation naturally describes the change of the GRA's reservation amount, but it does not allow the client to calculate the average speed of resource change over previous negotiation rounds.

To calculate the average speed of this change over multiple rounds, we have to use the estimation with the denominator which does not vary over time. So, this denominator is chosen to be $G_{i,0}^{max} - G_i^{min}$. According to Formula (3.20), $\delta_{i,k}/100\% = \left(G_{i,k}^{max} - G_{i,k}^{max'}\right) / \left(G_{i,k}^{max'} - G_i^{min}\right)$, where the expected GRA's reservation value in

round $k$ is equal to the actual reservation value in round $k-1$, i.e. $G_{i,k}^{max'} = G_{i,k-1}^{max}$. In other words, the client expected in round $k-1$ that the GRA's reservation value does not change in round $k$. Assuming $\Delta G_{i,k}^{max} = G_{i,k}^{max} - G_{i,k-1}^{max}$ and $\delta_{i,k}/100\% = \bar{\delta}_{i,k}$, then

$$\begin{aligned}
\bar{\delta}_{i,1} &= \frac{\Delta G_{i,1}^{max}}{\left(G_{i,0}^{max} - G_i^{min}\right)}, \\
\bar{\delta}_{i,2} &= \frac{\Delta G_{i,2}^{max}}{\left(G_{i,0}^{max} + \Delta G_{i,1}^{max} - G_i^{min}\right)}, \\
&\ldots, \\
\bar{\delta}_{i,m} &= \frac{\Delta G_{i,m}^{max}}{\left(G_{i,0}^{max} + \sum_{j=1}^{m-1} \Delta G_{i,j}^{max} - G_i^{min}\right)},
\end{aligned} \tag{3.30}$$

where $m$ is the total number of negotiation rounds. Applying arithmetic operations, Formula (3.30) can be also presented as below.

$$\begin{aligned}
\frac{\Delta G_{i,1}^{max}}{\left(G_{i,0}^{max} - G_i^{min}\right)} &= \bar{\delta}_{i,1}, \\
\frac{\Delta G_{i,2}^{max}}{\left(G_{i,0}^{max} - G_i^{min}\right)} &= \bar{\delta}_{i,2} \times (1 + \bar{\delta}_{i,1}), \\
\frac{\Delta G_{i,3}^{max}}{\left(G_{i,0}^{max} - G_i^{min}\right)} &= \bar{\delta}_{i,3} \times \left(1 + \bar{\delta}_{i,1} + \bar{\delta}_{i,2} + \bar{\delta}_{i,1}\bar{\delta}_{i,2}\right),
\end{aligned} \tag{3.31}$$

and so on. It has to be noted that in Formula (3.31), each next equation contains previous one e.g., the second equation has $\bar{\delta}_{i,1}$. The third equation has the sum of $\bar{\delta}_{i,1}$ and $\bar{\delta}_{i,2} \times (1 + \bar{\delta}_{i,1})$. Consequently, each next equation is composed of the previous ones. Therefore, if we assume that $\alpha_{i,k} = \Delta G_{i,k}^{max} / \left(G_{i,0}^{max} - G_i^{min}\right)$, then Formula (3.31) can be described in a different way.

$$\begin{aligned}
\alpha_{i,1} &= \bar{\delta}_{i,1}, \\
\alpha_{i,2} &= \bar{\delta}_{i,2} \times (1 + \alpha_{i,1}), \\
\alpha_{i,3} &= \bar{\delta}_{i,3} \times (1 + \alpha_{i,1} + \alpha_{i,2}), \\
&\ldots, \\
\alpha_{i,m} &= \bar{\delta}_{i,m} \times \left(1 + \sum_{j=1}^{m-1} \alpha_{i,j}\right).
\end{aligned} \tag{3.32}$$

The change of the GRA's reservation amount of resource in the round $k$ can be calculated as in Formula (3.32). The sum of $\alpha_{i,k}$ over multiple rounds provides the client with the overall direction and average speed of resource availability change.

#### 3.3.3.2 Prediction of the Outcome

A client intends to predict round $k_x$ in which an agreement would be reached with the GRA if the client's greediness was $\beta^c_{i,k}$ from the current round onwards. To estimate this round, a client assumes that the GRA's reservation amount of resource $G^{max}_{i,k}$ and the level of greediness $\beta^{gra}_{i,k}$ do not change in future rounds, because a client is not aware when resource availability may change and on which amount, nor how much those changes affect the GRA's proposals as long as we have only estimates of the GRA's negotiation parameters. Other reason to assume "no changes" in future is that a client focuses only on the sequential order of the predicted rounds for the different client tactics, but not their absolute values (this is explained in Section 3.3.3.3).

Assume that $t^c_{dl} = t^{gra}_{dl}$ and $\beta^{gra}_{i,k}$ is estimated as in Formula (3.15). If $\beta^{gra}_{i,k}$ is calculated and a client assumes that the GRA applies the time-dependent strategy of Faratin et al. [17], then $G^{max}_{i,k} - G^{min}_i = (t^c_{dl}/k)^{\beta^{gra}_{i,k}} \left(\hat{r}^{gra}_{i,k} - G^{min}_i\right)$. In those cases when $\beta^{gra}_{i,k}$ cannot be estimated because of the changes in resource availability, we assume that $\beta^{gra}_{i,k} = \beta^{gra}_{i,k-1}$. Following a discussion mentioned above, Equations (3.10), (3.11) for round $k_x$ can be presented as below.

$$\begin{aligned} \hat{r}^c_{i,k_x} &= R^{max}_i + \left(\frac{k_x}{t^c_{dl}}\right)^{\beta^c_{i,k}} \left(R^{min}_i - R^{max}_i\right), \\ \hat{r}^{gra}_{i,k_x} &= G^{min}_i + \left(\frac{k_x}{t^c_{dl}}\right)^{\beta^{gra}_{i,k}} \left(\frac{t^c_{dl}}{k}\right)^{\beta^{gra}_{i,k}} \left(\hat{r}^{gra}_{i,k} - G^{min}_i\right). \end{aligned} \tag{3.33}$$

Considering $\hat{r}^c_{i,k_x} = \hat{r}^{gra}_{i,k_x}$, Equation (3.34) is derived from Formula (3.33).

$$\left(\frac{k_x}{t^c_{dl}}\right)^{\beta^c_{i,k}} \frac{R^{max}_i - R^{min}_i}{R^{max}_i - G^{min}_i} + \left(\frac{k_x}{k}\right)^{\beta^{gra}_{i,k}} \frac{\hat{r}^{gra}_{i,k} - G^{min}_i}{R^{max}_i - G^{min}_i} - 1 = 0. \tag{3.34}$$

where $k_x$ is numerically calculated from Equation (3.34) using Newton tangent method.

#### 3.3.3.3 Evaluation Function

A client has to evaluate whether a particular level of greediness will lead to the best possible outcome. The best outcome can be the largest amount of resource or the most reachable amount of resource among the predicted amounts for the different levels of greediness, which are generated with various combinations of uncertainty intervals. A reachability of an agreement depends on the risk of resource exhaustion, i.e. the smaller this risk, the more reachable this agreement is considered by a client. Our *heuristic evaluation function* $Eval_{i,k}(k_x)$ estimates the risk of resource exhaustion every negotiation round $k$ for task $i$ by considering the overall direction and the average speed of resource dynamism inferred from the previous negotiation rounds. This function is based on the two principles:

1. The longer negotiation, the larger amount of resource can be obtained, if the GRA's reservation value does not decrease substantially over time;
2. The shorter negotiation, the smaller risk of resource exhaustion during negotiation.

In our work, a client submits its resource requirements to the GRA and the GRA replies with a proposal. In this way, the GRA makes a proposal each negotiation round, while a client replies with a counter-proposal. Therefore, the longer negotiation, the closer GRA's proposed amount of resource to its reservation amount (i.e. the largest amount that the client can obtain). In most cases, the client intends to obtain the larger amount of resource. However, if resource availability is mostly decreasing during negotiation, the risk of resource exhaustion becomes higher with every next negotiation round. Especially, this risk increases if the speed of decrease of resource availability is high. Therefore, the shorter negotiation is considered to reduce the possibility of resource exhaustion during negotiation.

Considering the two principles mentioned above, a client maximises the evaluation function $max_{k_x}(Eval_{i,k}(k_x))$ to find the longest or the shortest negotiation each round $k$ based on the expected round of agreement $k_x$. Thus, the client chooses that level of greediness $\beta^c_{i,k}$ which corresponds to the longest or the shortest negotiation. In this way, the evaluation function $Eval_{i,k}(k_x)$ has two summands, one of which indicates the longest negotiation and another one indicates the shortest negotiation when a maximum of $Eval_{i,k}(k_x)$ is estimated. If the risk of resource exhaustion is low, then the summand

which corresponds to the longest negotiation is maximised, while another summand is equal to zero and vice verse. To indicate which summand has to be maximised, we use the Heaviside step function $\theta(\cdot)$, i.e.

$$\theta(Cm_{i,k}) = \begin{cases} 1, & Cm_{i,k} \geq 0 \\ 0, & Cm_{i,k} < 0 \end{cases} \tag{3.35}$$

where

$$Cm_{i,k} = \gamma \sum_{j=1}^{k} \alpha_{i,j} + \frac{k}{t_{dl}^{c}} \tag{3.36}$$

where $\gamma$ is the level of sensitivity of the client in respect to the speed of resource change and it is considered to be equal to one. The idea of Equation (3.36) is that if the GRA's reservation amount of resource $G_{i,k}^{max}$ decreases with such average speed $(\gamma/k)\sum_{j=1}^{k}\alpha_{i,j}$ that it may reach the minimum amount $G_{i}^{min}$ earlier than both agents reach negotiation deadline[4], then resources can be exhausted during negotiation and this speed is considered to be high (see Figure 3.9).

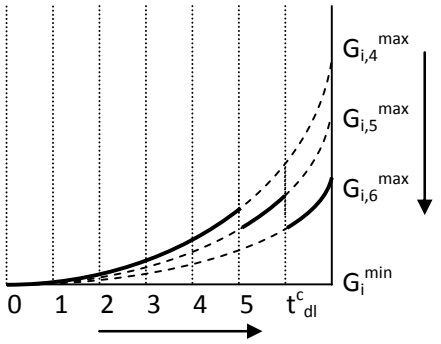



FIGURE 3.9: The GRA's proposals over time

Figure (3.9) shows the proposals of the GRA over multiple rounds. The arrows depict the direction of the change of the GRA's reservation amount and negotiation rounds. When $G_{i,k}^{max}$ becomes smaller than $G_{i}^{min}$, resources are considered to be exhausted. According to this idea, $Cm_{i,k} < 0$ when $G_{i,k}^{max}$ mostly decreases with the average speed $\left|(\gamma/k)\sum_{j=1}^{k}\alpha_{i,j}\right|$ higher than $1/t_{dl}^{c}$. $Cm_{i,k} \geq 0$ when $\left|(\gamma/k)\sum_{j=1}^{k}\alpha_{i,j}\right|$ mostly increases with any speed or mostly decreases with the average speed $\left|(\gamma/k)\sum_{j=1}^{k}\alpha_{i,j}\right|$ lower than

[4]In our current work, this deadline is the same for a client and the GRA.

$1/t^c_{dl}$. Finally, the evaluation function is presented in Formula (3.37).

$$Eval_{i,k}(k_x) = \left(\frac{k_x}{t^c_{dl}}\right)\theta\left(Cm_{i,k}\right) + \left(\frac{t^c_{dl} - k_x}{t^c_{dl}}\right)\left(1 - \theta\left(Cm_{i,k}\right)\right), \tag{3.37}$$

where $k_x/t^c_{dl}$ and $\left(t^c_{dl} - k_x\right)/t^c_{dl}$ are the relative numbers of rounds before and after the expected round of agreement $k_x$ respectively. If a client maximises the first summand, the longest negotiation is chosen. If a client maximises the second summand, the shortest negotiation is chosen.

#### 3.3.3.4 Variation of Uncertainty Intervals

In our algorithm, the variation of uncertainty intervals has several constraints to avoid an ambiguity in the resulting client's greediness $\beta^c_{i,k}$ and to reduce the amount of computations. That is,

1. To avoid an excessive uncertainty in the input and output membership functions (see Sections 3.3.2.1 and 3.3.2.2), the one intersection should be created by only two adjacent fuzzy sets. Therefore, we do not consider the case when more than two fuzzy sets can share the same intersection.

2. To avoid the case when an input (e.g. $\beta^c_{i,k}$) or output crisp value (e.g. $\eta_{i,k}\%$) does not belong to any fuzzy set of the corresponding membership functions, every two adjacent fuzzy sets should intersect. In this way, this crisp value cannot have the degree of membership for all fuzzy sets equal to zero.

3. To avoid unnecessary computations, the uncertainty interval of a particular fuzzy set is varied only, if this fuzzy set affects the resulting client's greediness $\beta^c_{i,k}$ in round $k$. Therefore, we consider only those fuzzy sets in which the degree of membership of the input or output crisp values for the corresponding membership functions is non-zero.

It also has to be noted that in our input membership functions (see Section 3.3.2.1), we vary only the uncertainty intervals of middle fuzzy sets, i.e. Z and IND, as a variation of other input uncertainty intervals would cause an inconsistency in our model. For example, the GEN fuzzy set represents the set of the client's generous tactics that belong to the interval $[0, 1]$ (see Formula (3.12)) and, therefore, it cannot have an

uncertainty interval other than $[0, 1]$. However, the upper limit of the GR and GEN fuzzy sets, where the degree of membership of a crisp value equals to one, is varied together with the sides of IND fuzzy set and the same variation is applied to the D and I fuzzy sets synchronously with the sides of Z fuzzy set. Now, we explain in detail our algorithm of uncertainty intervals' variation based on our output membership function (see Section 3.3.2.2) as an example.

Considering our description of the output membership function in Section 3.3.2.2 and the constraints mentioned above, we vary each side of the fuzzy set, which is depicted as a triangle, in Figure 3.5 in the interval which length is equal to 50. The sides are varied in the interval from the top of one triangle to the top of another one, while the tops of those triangles are not varied because they refer to the one degree of membership of $\eta_{i,k}$. In other words, the client is absolutely certain that they belong to the corresponding fuzzy sets. For example, the right side of the LD fuzzy set is varied in the interval $]-100, -50]\,\%$ to ensure that it intersects only with the MD fuzzy set, i.e. from a position which is close to the top of LD fuzzy set towards the top of MD fuzzy set.

Our algorithm divides each interval of variation (e.g. 50 for the output membership function) on several segments which denote possible positions of the varied sides. For instance, the interval $]-100, -50]\,\%$ can be divided on two segments $]-100, -75]\,\%$ and $]-75, -50]\,\%$. In this case, the right side of LD fuzzy set can be set in the positions: $-75$ and $-50$. If we divide each interval in two segments, this algorithm is called the algorithm of variation with *two steps*. The larger number of segments, the larger number of variations of the uncertainty intervals we can take into account. When we move the sides of triangles which should make an intersection, they are moved from the closest position to the top position of its triangle towards the top position of the adjacent triangle (or trapezium), considering only combinations which comply with the constraints mentioned above. For example, if we vary the left side of MD fuzzy set and the right side of LD fuzzy set, a client considers such combinations of these sides' positions as (-75, LD), (-100, MD); (-50, LD), (-75, MD); and (-50, LD), (-100, MD). That is, the position of the MD side always has to be smaller by value than the position of the LD side to ensure their intersection. The two sides also cannot be in the same position at the same time because it is against the second constraint e.g., (-75, LD), (-75, MD).

Our algorithm of negotiation between a client and the GRA, emphasising on decision-making for a client, is presented in Algorithm 3.1, where a variation of uncertainty

intervals is depicted schematically in lines 17 and 20 for the input and output fuzzy sets respectively. In our implementation, all combinations are simulated with 12 nested

**Algorithm 3.1** Negotiation algorithm between a client and the GRA

1: **for** each task $i$ **do**
2: {repeat for each negotiation round $k$}
3: **repeat**
4: $t = k \times \tau^{elem}$ {Current time}
5: **if** $G_{i,k}^{max} < G_i^{min}$ **then**
6: the GRA sends a message REJECT {Resource exhausted}
7: **else**
8: **if** $G_{i,k}^{max} > R_i^{max}$ **then**
9: the GRA sends a proposal $Pr_{i,k}^{gra}$ and $G_{i,k}^{max} = R_i^{max}$, or a message ACCEPT {Section 3.2.1.2}
10: **else**
11: the GRA sends a proposal $Pr_{i,k}^{gra}$ or a message ACCEPT {Section 3.2.1.2}
12: **end if**
13: **end if**
14: {the **client**'s decision making process}
15: **if** $k \geq 2$ **then**
16: predicts $k_x$ for $\beta_{i,k}^{c}$ {Equation (3.34)}
17: calculates $Eval_{max} = Eval_{i,k}(k_x)$ {Equation (3.37)}
18: **for** the combinations of input uncertainty intervals **do**
19: fuzzifies $\beta_{i,k}^{c}$ and $\delta_{i,k}$ {Section 3.3.2.1}
20: infers which fuzzy sets are non-zero {Section 3.3.2.4}
21: **for** the combinations of output uncertainty intervals **do**
22: defuzzifies $\eta_{i,k}$ {Section 3.3.2.5}
23: calculates $\beta_{i,k}^{c'} = \beta_{i,k}^{c}(1 + \eta_{i,k}/100\%)$
24: predicts $k_x$ for $\beta_{i,k}^{c'}$ {Equation (3.34)}
25: calculates $Eval_{i,k}(k_x)$ {Equation (3.37)}
26: **if** $Eval_{i,k}(k_x) > Eval_{max}$ **then**
27: $Eval_{max} = Eval_{i,k}(k_x)$
28: $\beta_{i,k}^{c'}$ is chosen
29: **end if**
30: **end for**
31: **end for**
32: **end if**
33: sends a counter-proposal $Pr_{i,k}^{c}$ or a message ACCEPT {Section 3.2.1.2}
34: $k = k + 1$
35: **until** $(t > t_{dl}^{c})$ or $(t > t_{dl}^{gra})$
36: **end for**

FOR-loops, where each of them represents one side of a particular fuzzy set. So, the

combination of these loops give us one combination of uncertainty intervals of the non-zero fuzzy sets. The loops have such start and end conditions that they iterate just once for zero fuzzy sets. The sides of zero fuzzy sets are automatically set into the furthest allowable positions from the tops of their respective triangles to allow the adjacent non-zero fuzzy sets to check all possible combinations of their uncertainty intervals without a risk of violating the constraints. For example, if the MD fuzzy set is zero, its left side's position will be $-100$ and its right side's position will be 0. That is, the start and end conditions of the loops are programmed in such way that it allows the client to avoid the combinations of uncertainty intervals of the non-zero fuzzy sets which are against our constraints.

In this algorithm, the GRA accepts a client's proposal in round $k$ if $\hat{r}^{gra}_{i,k} > \hat{r}^{c}_{i,k-1}$, while a client accepts the GRA's proposal in round $k$ if $\hat{r}^{c}_{i,k} < \hat{r}^{gra}_{i,k}$. In other words, if the negotiator's would-be proposal (i.e. the amount of resource) is worse than the last opponent's proposal, than the negotiator accepts its last opponent's proposal. Here, if the deadline of negotiation is reached, each negotiator can send the last proposal (its reservation value), and if an agreement is not reached, a message REJECT is sent by either party.

**Example for Algorithm 3.1.**

This example shows a process of negotiation between a client and the GRA. It demonstrates how the client's level of greediness changes if the GRA's reservation value does not change or substantially decreases. Here, a client considers an algorithm for varying uncertainty intervals (see Algorithm 3.1) with two steps. The client's minimum and maximum resource requirements are equal to 58.4793 and 292.397 respectively, while the GRA's reservation and aspiration resource amounts are equal to 275.375 and 70.0388 at the beginning of negotiation.

**Round: 0**

The GRA's reservation value: 275.375; The GRA' s level of greediness: 1.

The client's level of greediness: 1.99782.

The GRA's proposal: 70.0388; The client's proposal: 292.397.

**Round 1:**

The GRA's reservation value:: 275.375; The GRA' s level of greediness: 1.

The client's level of greediness: 1.99782.

The GRA's proposal: 72.0922; The client's proposal: 292.373.

A client does not change its level of greediness during the initial two rounds, because it needs at least two rounds in order to estimate the GRA's level of greediness and the change in the increment of the GRA's reservation value (see Sections 3.3.1.1 and 3.3.1.2). Starting from the second round, a client considers the number of fuzzy sets' combinations and chooses the most appropriate level of greediness, according to the largest output of evaluation function. If the GRA's reservation value does not change, then the client becomes more greedy as demonstrated in round 2.

**Round 2:**

The GRA's reservation value: 275.375; The GRA' s level of greediness: 1.

The client's level of greediness: 1.99782.

The GRA's proposal: 74.1455.

The client's estimation of the change in the GRA's reservation value: 0; The client's estimation of the GRA's level of greediness: 1.

The output of evaluation function, 0.630121, for the current client's level of greediness.

**Uncertainty intervals' variation:**

| The value of evaluation function | The client's expected level of greediness |
|---|---|
| 0.630121 | 1.99782 |
| **0.645225** | **2.18326** |
| 0.613275 | 1.81238 |
| 0.630121 | 1.99782 |
| 0.644184 | 2.16983 |
| 0.614561 | 1.82581 |
| 0.630121 | 1.99782 |
| ... | ... |

**The chosen client's level of greediness: 2.18326.**

The client's proposal: 292.351.

If the GRA's reservation value changes, the client is unable to estimate this change in the same round. Therefore, the client assumes that the GRA's reservation value has not changed, and the GRA's level of greediness is the same as in the previous round. This situation is demonstrated in round 3. As a result, the client chooses the larger level of greediness in this round, despite a decrease in the GRA's reservation value. However, the client is able to estimate the GRA's level of greediness and the change in the GRA's reservation value in round 4.

**Round 3:**

The client's level of greediness: 2.18326.

The GRA's reservation value: 262.584; The GRA's level of greediness: 1.21872.

The GRA's proposal: 72.7215.

The client's estimation of the change in the GRA's reservation value: 0; The client's estimation of the GRA's level of greediness: 1.

**The chosen client's level of greediness: 2.39286.**

The client's proposal: 292.344.

When the client finds out that the GRA's reservation value has decreased significantly enough in order to rise a risk of resource exhaustion, the client's level of greediness decreases as demonstrated in round 4. Note that the number of fuzzy sets' combinations considered by a client in round 4 is larger than in the previous round, because this number depends on how many output fuzzy sets are non-zero. Hence, we omitted some of the outputs of evaluation functions due to the large number of those combinations.

**Round 4:**

The client's level of greediness: 2.39286.

The GRA's reservation value: 262.584; The GRA's level of greediness: 1.21872.

The GRA's proposal: 73.848.

The client's estimation of the change in the GRA's reservation value: -6.22907%; The client's estimation of the GRA's level of greediness: 1.21872

The output of evaluation function, 0.303491, for the current level of greediness.

**Uncertainty intervals' variation:**

| The value of evaluation function | The client's expected level of greediness |
|---|---|
| **0.340634** | **1.89029** |
| 0.320564 | 2.14397 |
| 0.338393 | 1.91676 |
| 0.315857 | 2.2093 |
| ... | ... |

**The chosen client's level of greediness: 1.89029.**

The client's proposal: 291.864.

...

**Round: 37**

The client's level of greediness: 0.0525735.

The GRA's reservation value: 166.354; The GRA's level of greediness: 5.51316.

The GRA's proposal: 70.4398.

The client's would-be proposal: 67.6995.

The negotiation ends in round 37 with the client obtaining the amount of resource, 70.4398, which is the last best proposal of the GRA, i.e. the client has accepted the GRA's proposal. In our model, the GRA sends its proposal every negotiation round and the client replies with a counter-proposal (if applicable). The GRA may decide not to send a proposal, but to accept or reject the last client's counter-proposal instead.

TABLE 3.3: List of notation for Section 3.3

| Symbol | Notation |
|---|---|
| $k$ | An identifier of the round of negotiation, where $k \in \mathbb{N}$. |
| $\hat{r}_{i,k}^{gra}$, $\hat{r}_{i,k}^{gra'}$ | The proposed by the GRA (expected to be proposed) amount of resource $\hat{r}_{i,k}^{gra} \geq 0$ ($\hat{r}_{i,k}^{gra'} > 0$) in round $k$ for task $i$, where $k, i \in \mathbb{N}$. |
| $\beta_{i,k}^{gra}$, $\beta_{i,k}^{gra'}$ | The actual or expected by a client GRA's level of greediness respectively in round $k$ for task $i$, where $k, i \in \mathbb{N}$. |

Continued on the next page

Continued from the previous page

| Symbol | Notation |
|---|---|
| $\Delta\hat{r}_{i,k}^{gra}$, $\Delta\hat{r}_{i,k}^{gra'}$ | The difference between the proposed by the GRA (expected) amount of resource $\hat{r}_{i,k}^{gra} \geq 0$ ($\hat{r}_{i,k}^{gra'} > 0$) in round $k$ and its initial proposal $G_i^{min} > 0$ respectively, where $k, i \in \mathbb{N}$. |
| $\Delta\hat{r}_{i,k}^{max}$, $\Delta\hat{r}_{i,k}^{max'}$ | The difference between the actual GRA's reservation value $G_{i,k}^{max} > 0$ (or expected $G_{i,k}^{max'} > 0$) and the GRA's aspiration value $G_i^{min} > 0$ in round $k$, where $k, i \in \mathbb{N}$. |
| $\delta_{i,k}$ | The client's estimate of the change in the increment of the GRA's reservation value in round $k$ for task $i$, where $k, i \in \mathbb{N}$. |
| $\eta_{i,k}\%$ | The percentage of the client's level of greediness $\beta_{i,k-1}^c$ in round $k-1$ on which $\beta_{i,k-1}^c$ has to be increased or decreased in order to calculate its level of greediness $\beta_{i,k}^c$ in round $k$, where $k, i \in \mathbb{N}$. |
| $\mu_A(x)$ | A generic representation of the membership function, which produces the values from the interval $[0, 1]$, of the variable $x$ in fuzzy set $A$. |
| $\mu_l^F\left(\delta_{i,k}, \beta_{i,k}^c\right)$ | The resulting input membership function, which produces a minimum (conjunction) of the membership functions $\mu_{X'}(\delta_{i,k})$ and $\mu_{Y'}\left(\beta_{i,k}^c\right)$ where $X' = D\|Z\|I$ and $Y' = GEN\|IND\|GR$, according to control rule $l$ which refers to a particular output fuzzy set $F = LD\|MD\|Z\|MI\|I$, where $k, i, l \in \mathbb{N}$. |
| $\mu_F^l(\eta_{i,k})$ | The truncated membership function $\mu_F(\eta_{i,k})$ for the output fuzzy set $F$, according to control rule $l$, where $k, i, l \in \mathbb{N}$. |
| $\phi(\cdot)$ | An implication function, which truncates the membership function $\mu_F(\eta_{i,k})$, applying a minimum operator to $\mu_F(\eta_{i,k})$ and $\mu_l^F\left(\delta_{i,k}, \beta_{i,k}^c\right)$, where $k, i, l \in \mathbb{N}$. |
| $\varphi(\cdot)$ | An aggregation function, which aggregates all truncated membership functions $\left[\mu_{F_j}^l(\eta_{i,k})\right]_{CR,NS}$ for all fuzzy rules $CR$ and output fuzzy sets $NS$, where $k, i, l, j, CR, NS \in \mathbb{N}$. |
| $COG(\mu_{res}(\eta_{i,k}))$ | A function which calculates the center of gravity of the area under the inferred through implication and aggregation output membership function $\mu_{res}(\eta_{i,k})$, where $k, i \in \mathbb{N}$. |

Continued on the next page

Continued from the previous page

| Symbol | Notation |
|---|---|
| $\Delta G_{i,k}^{max}$ | The difference between the GRA's reservation value $G_{i,k}^{max}$ in round $k$ and this value $G_{i,k-1}^{max}$ in the previous round $k-1$, where $k, i \in \mathbb{N}$. |
| $\alpha_{i,k}$ | A ratio of the difference in the GRA's reservation value $\Delta G_{i,k}^{max}$ in round $k$ to its initial negotiation interval $G_{i,0}^{max} - G_i^{min}$, where $k, i \in \mathbb{N}$. |
| $Eval_{i,k}(k_x)$ | An evaluation function which is used by a client to choose the best tactic in order to avoid a negotiation failure due to the possible resource exhaustion during negotiation every round $k$ for task $i$, where $Eval_{i,k}(k_x) \in [0,1]$, $k, i, k_x \in \mathbb{N}$. |
| $Cm_{i,k}$ | The component of evaluation function $Eval_{i,k}(k_x)$ which produces negative values, if there is a risk of resource exhaustion, and positive values otherwise, where $Cm_{i,k} \in \mathbb{R}$, $k, i \in \mathbb{N}$. |
| $\theta(\cdot)$ | The Heaviside step function which is equal to zero, if its argument is negative, and to one, if its argument is positive or equal to zero. |

## 3.4 Results Discussion

To test our adaptive strategy, we conducted two experiments for the low and high-speed resource dynamism. The low speed of resource change denotes that the GRA's reservation amount of resource $G_{i,k}^{max}$ for task $i$ can change at most on 5% of the corresponding client's maximum amount of resource $R_i^{max}$ per round, while the high speed means the change up to 20%. Those quantitative choices aim to show two substantially distinctive cases, where the GRA's reservation value changes on smaller or larger amounts per negotiation round respectively. We also evaluate our strategy for a number of other resource deviations in the following chapters. The minimum resource change denotes 0% and the uniform distribution is considered to generate these changes. The resource dynamism is modelled by the probability of tendency, i.e. the probability that the direction of the next resource change is the same as the direction of the previous one. In this way, the most tendentious resource dynamism is when this probability is equal to 1, i.e. each next resource change always follows the direction of the previous one. The most random resource dynamism is when this

probability is equal to 0, i.e. each next resource change reverses the direction of the previous one.

Each experiment considers at most 100 rounds of negotiation for each task, where the averaged client utility (see Section 3.2.1.3) is calculated over 30 runs. The maximal number of negotiation rounds has been chosen experimentally, because the smaller number of rounds does not show tendencies in the resource availability and, as a result, the full potential of our strategy, while the larger number of rounds does not introduce new effects, but increases the running time of algorithm. The number of tasks is 100 and the initial greediness of the client is 1.99, while the GRA is 1.0. That is, the GRA is considered to be indifferent in respect to all clients before negotiation, while a client is considered to be greedy in respect to the GRA because it intends to obtain the larger amount of resource. As long as a client needs at least two rounds to estimate the GRA's negotiation parameters, it chooses a level of greediness from the start of negotiation which allows its utility to fall only on $10^{-4}$ per concession per negotiation round. This level of greediness for a client is also chosen, because it is close to the border of the different tactics (see Formula (3.12)), i.e. a client can switch into generous or indifferent behaviour faster if necessary, which means a rational choice for the client when it does not have prior knowledge about resource availability. To explain the obtained utilities, we also calculated the number of successful negotiations among 100 tasks, and the distribution of resources among those tasks which are categorised in four groups. That is, the tasks which obtained $0-25\%$, $25-50\%$, $50-75\%$ and $75-100\%$ resource amounts of their maximum amounts $R_i^{max}$.

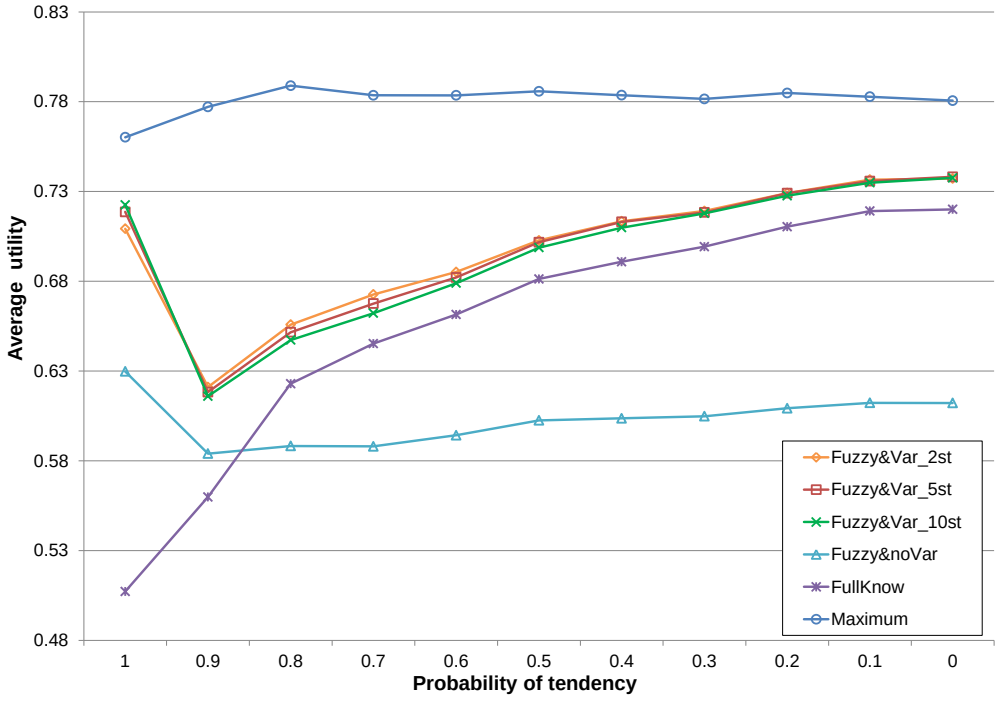
To evaluate our algorithm of varying uncertainty intervals, we compare our adaptive negotiation strategy with two ('Fuzzy&Var_2st'), five ('Fuzzy&Var_5st') and ten steps ('Fuzzy&Var_10st') (see Section 3.3.3.4). The strategies with the larger number of steps allow a client to consider the larger number of combinations of uncertainty intervals. We also compare our new adaptive strategy to a strategy ('Fuzzy&noVar') [25], which responds on the changes in resource availability in the current round, but it does not take into account the tendencies of these changes over time. In our evaluation, we also consider the negotiation strategy ('FullKnow') proposed by Sim et al. [28], where the client has a full knowledge about the negotiating parameters of its opponent. We adopted this strategy for the case when the negotiators' deadlines are equal and, thus, the best strategy for the client is that it uses the same reservation value as the GRA to ensure that this value will be obtained at the end of negotiation. However, this strategy does not take into account that resources can be exhausted during negotiation

and, therefore, the client may not obtain the GRA's reservation value. This strategy has been chosen for comparison, because it is a well-known strategy which shows an optimal outcome for the cases of no risk resource exhaustion, and it has also been expected to show a utility improvement for the cases of low risk resource exhaustion, compared to our strategies. Finally, we calculate the client utilities for all probabilities of tendency when the client obtains the maximum possible amounts of resources during negotiation ('Maximum'). This is an ideal scenario, i.e. any negotiation strategy is not expected to outperform 'Maximum'. The maximum possible amount of resource denotes the largest amount that the GRA proposed during negotiation (this amount is not necessarily its reservation value in our settings).

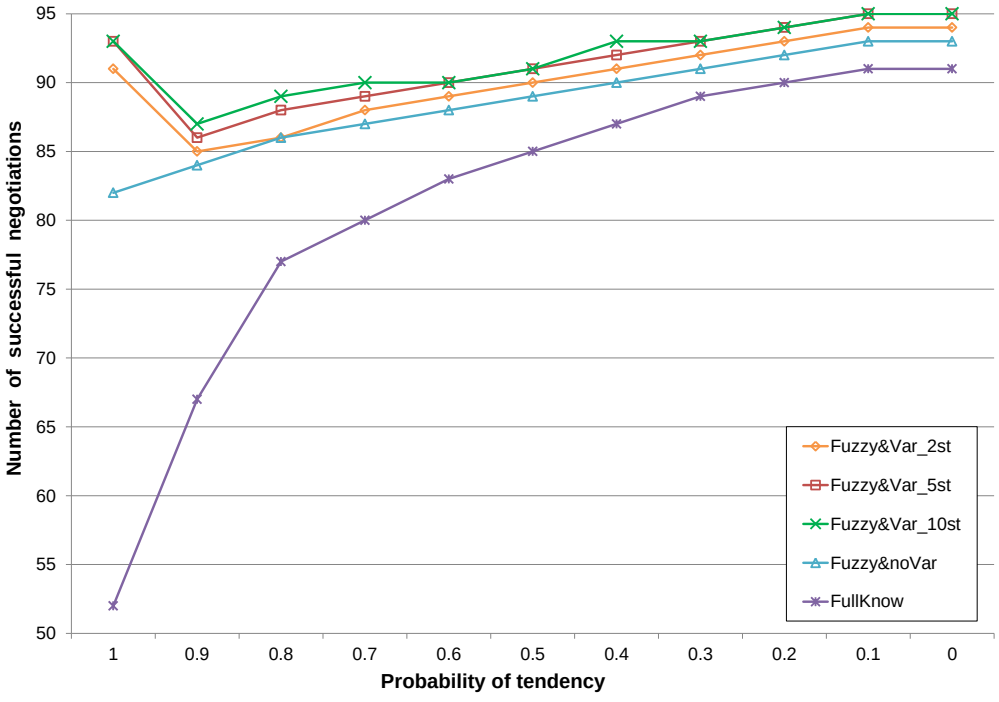
### 3.4.1 Low Speed Resource Dynamism

Figure 3.10 shows the averaged utilities and the number of successful negotiations for the low speed resource dynamism. In Figures 3.10(a) and 3.10(b), our adaptive strategies 'Fuzzy&Var_2st', 'Fuzzy&Var_5st' and 'Fuzzy&Var_10st' demonstrate the higher utilities and the larger number of successful negotiations than all other strategies. Although the difference in utilities for our strategy with the different number of steps 'Fuzzy&Var_2st', 'Fuzzy&Var_5st' and 'Fuzzy&Var_10st' is not significant, the larger number of steps leads to a slightly larger number of successful negotiations. The distribution of resources among tasks for the cases 'Fuzzy&Var_2st' and 'Fuzzy&Var_5st' in Figures 3.11(a) and 3.11(b) confirms that the usage of the larger number of combinations of uncertainty intervals compared to the cases with their smaller number does not lead to a significant change in utilities. This can be explained that the resource availability does not decrease or increase rapidly for the lower speeds of resource dynamism and the client is able to adapt correctly to the future changes even with the less choice of the levels of greediness, i.e. the smaller number of combinations of uncertainty intervals.

The other reason for this is that our evaluation algorithm (see Section 3.3.3.3) intends to choose the smallest or the largest level of greediness $\beta^c_{i,k}$ each negotiation round. Sometimes, the more extreme greediness can be calculated when the side of triangle is closer to its top position that may lead to the more significant decrease or increase of $\beta^c_{i,k}$ during negotiation. As a result, the larger number of segments, on which the interval of variation of a particular side is divided, leads to the closer positions of this side in respect to the top position of the triangle. When the resource availability decreases,




(a) Client Utility




(b) Number of successful negotiations

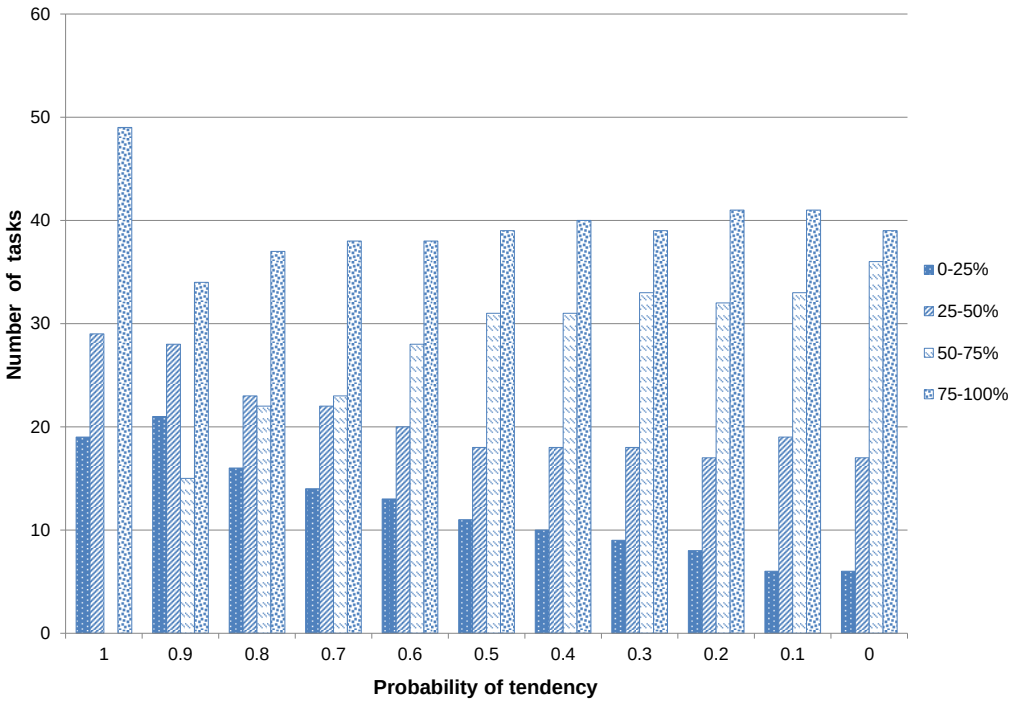
FIGURE 3.10: Client utilities for the low speed resource dynamism

$\beta^c_{i,k}$ may decrease more significantly for the case 'Fuzzy&Var_5st' than for the case 'Fuzzy&Var_2st'. Consequently, the number of successful negotiations increases for the case 'Fuzzy&Var_5st', increasing a bit the number of tasks in the range group $25-50\%$ compared to the case 'Fuzzy&Var_2st'. On the other hand, the case 'Fuzzy&Var_2st' has just a slightly larger number of tasks in the range group $75-100\%$, compared to the case 'Fuzzy&Var_5st', for the larger probabilities of tendency, caused by the less rapid change in the level of greediness which is beneficial for the lower speeds in the case of occasional resource availability decreases.

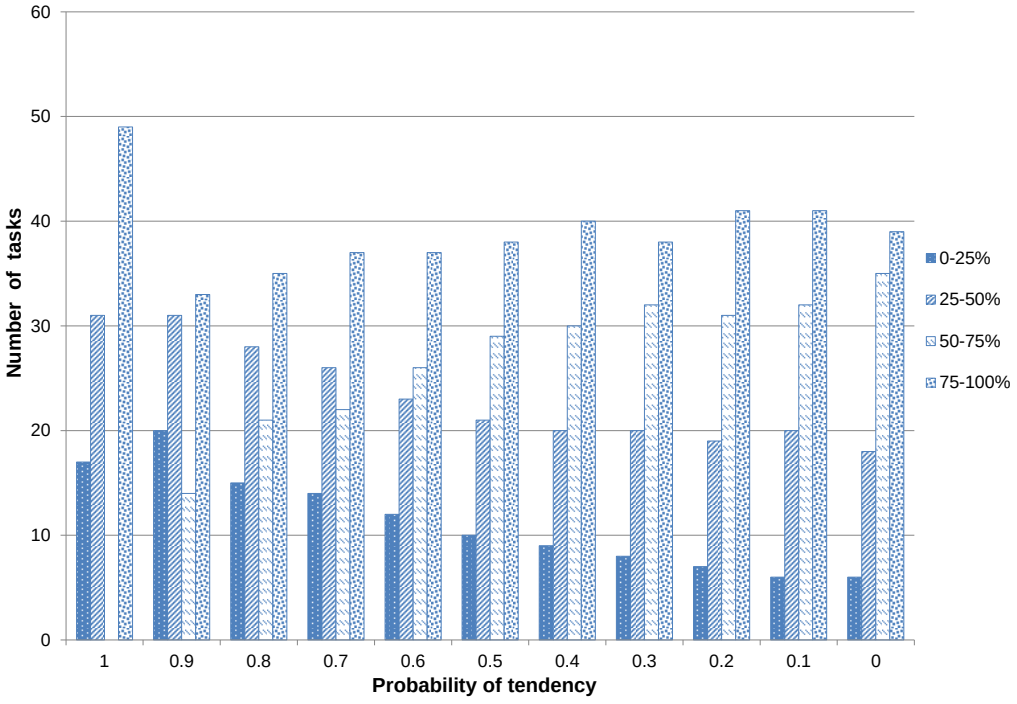
In Figure 3.10, the client utilities in the cases 'Fuzzy&Var_2st'and 'Fuzzy&Var_5st' for the probability of tendency 0.9 are significantly different from the utilities for other probabilities. So, when the probability of tendency is 1, it means that the resource availability changes in the same direction and the client is able to correctly predict the future changes based on the past changes. When the probability is equal to 0.9, it means that the resource availability mostly changes in one direction, but then it may reverse direction unexpectedly. Therefore, the client's estimations about the future changes will be wrong in this case and the client may not have time to adapt to the new changes. When the probability decreases towards 0, the level of consistency in resource changes with respect to the direction also decreases. Thus, the client is more able to estimate the future changes correctly on average.

The adaptive strategy 'Fuzzy&noVar' does not demonstrate a significant increase in utility over the different probabilities of tendency. This happens because this strategy responds on each increase of resource availability by increasing its greediness and on each decrease by decreasing its greediness without considering the overall direction and an average speed of resource dynamism. So, the client's greediness averages during negotiation and it also does not change when the resource availability is the same as in the previous negotiation round. In other words, the client's greediness does not change significantly over rounds, thus all tasks are mostly concentrated in the second and third groups (see Figure 3.12(a)). Moreover, the number of tasks in the first and the fourth groups is gradually decreasing towards the smaller probability of tendency.

The strategy with a full knowledge 'FullKnow' about an opponent shows the consequences of insensitivity in respect to the resource exhaustion. Therefore, the client's utility increases when the number of possible failed negotiations decreases, i.e. towards the probability of tendency 0. The number of successful negotiations for this strategy is the smallest compared to all other cases. However, this strategy shows that when

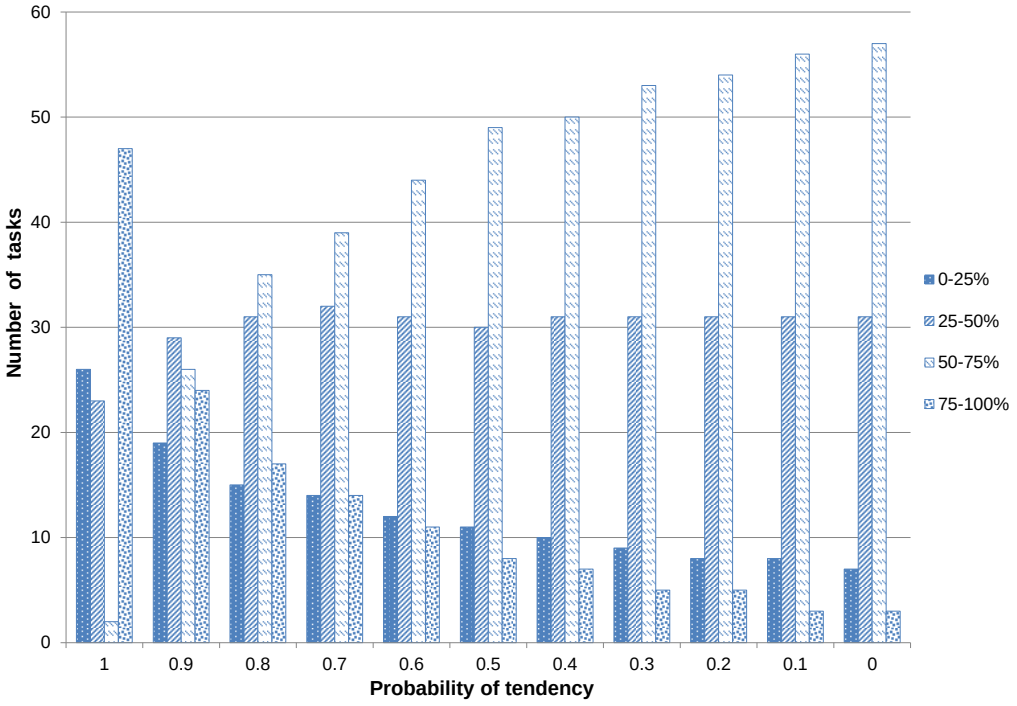



(a) Resource distribution for 'Fuzzy&Var_2st'

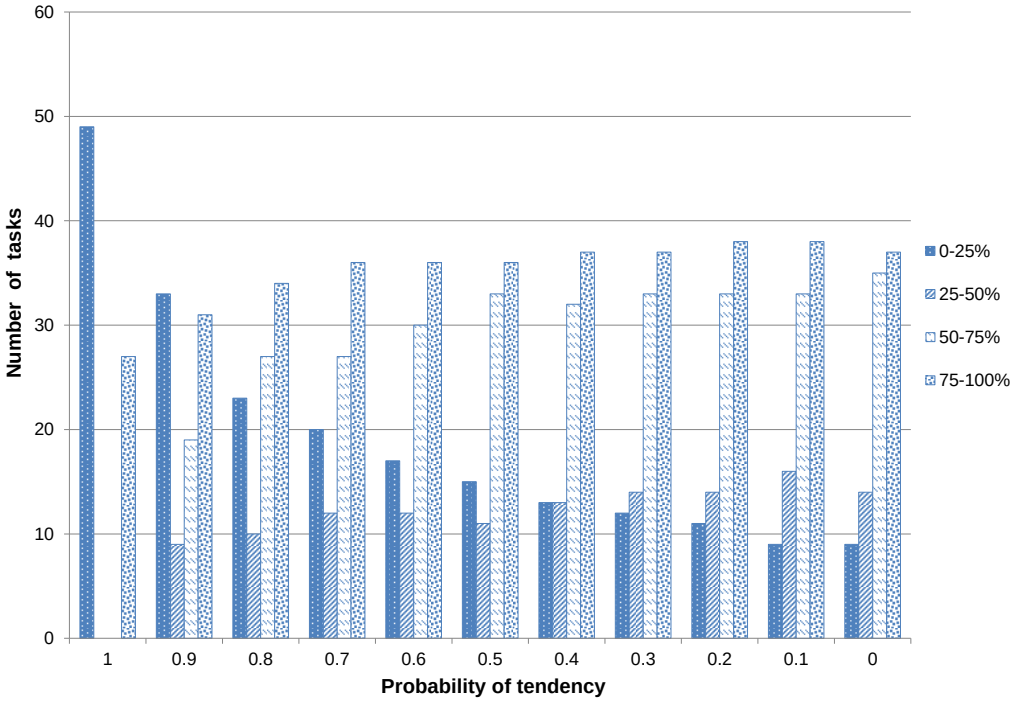



(b) Resource distribution for 'Fuzzy&Var_5st'

FIGURE 3.11: Resource distribution for the low speed resource dynamism for the cases 'Fuzzy&Var_2st' and 'Fuzzy&Var_5st'




(a) Resource distribution for 'Fuzzy&noVar'




(b) Resource distribution for 'FullKnow'

FIGURE 3.12: Resource distribution for the low speed resource dynamism for the cases 'Fuzzy&noVar' and 'FullKnow'

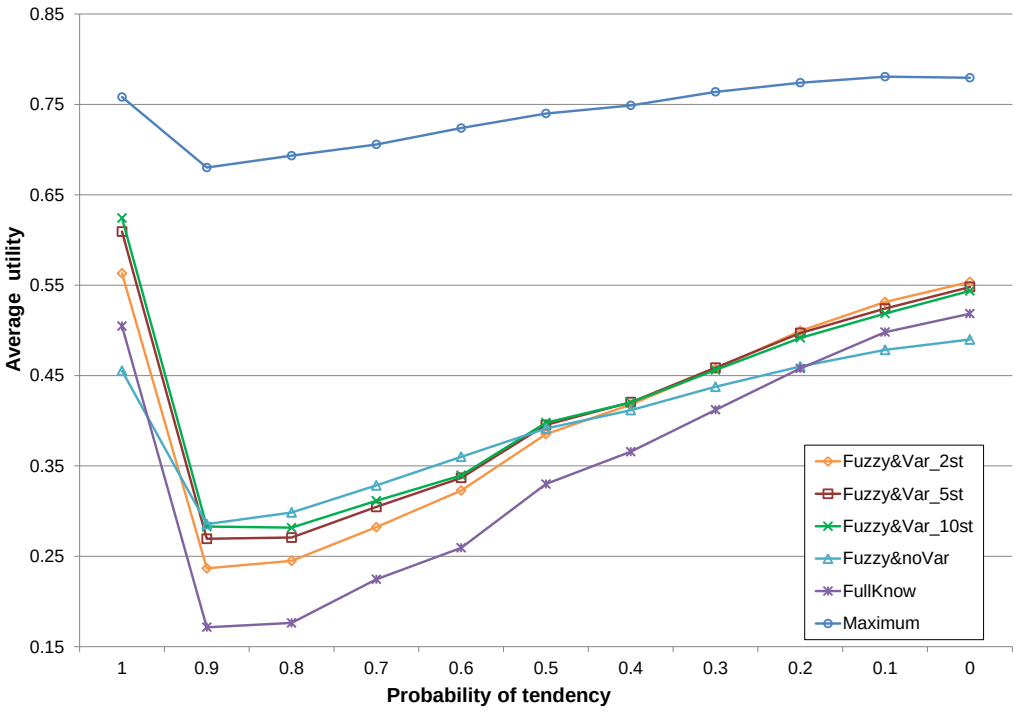
the number of failed negotiations decreases, the larger number of tasks moves to the range groups $50-75\%$ and $75-100\%$ (see Figure 3.12(b)). Thus, this strategy can be efficient when the resource exhaustion is not considered.

### 3.4.2 High Speed Resource Dynamism

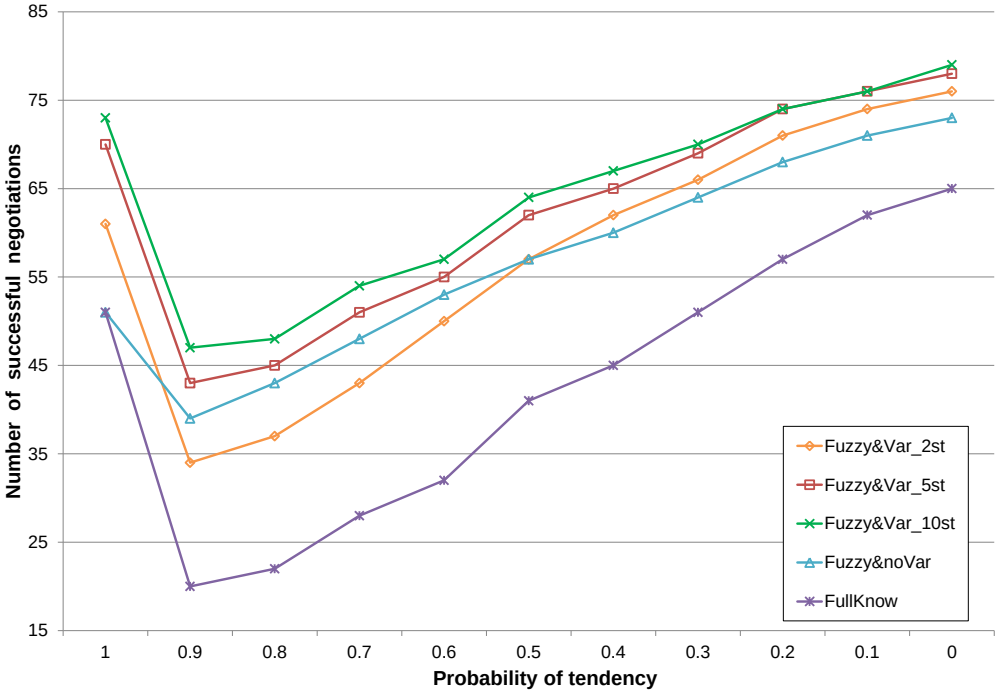
Figure 3.13 shows that the utilities (see Figure 3.13(a)) more significantly depend on the number of successful negotiations (see Figure 3.13(b)) for the higher speeds than for the lower speeds, because the possibility of resource exhaustion during negotiation rises. Thus, the difference between the utilities for the higher and lower probabilities of tendency (excluding probability 1) is more significant than for the lower speed discussed in the previous section. When the number of successful negotiations increases towards the probability of tendency 0, the utilities rise respectively. Our adaptive strategy for both cases 'Fuzzy&Var_2st' and 'Fuzzy&Var_5st' is outperformed by the strategy 'Fuzzy&noVar' in the interval of tendencies $[0.9, 0.6]$ because of the large unexpected resource fluctuations, which are more likely to occur within this interval. However, our strategy demonstrates approximately the same utility as 'Fuzzy&noVar' for the probability of tendency 0.5 and the higher utilities for all other probabilities, where our strategy is able to predict adequately future changes in resource availability.

Moreover, our strategy with ten steps 'Fuzzy&Var_10st' shows the larger number of successful negotiations than all other strategies. It also outperforms our strategies with two 'Fuzzy&Var_2st' and five steps 'Fuzzy&Var_5st' in the interval of tendencies $[0.9, 0.6]$ that allows us to believe in a potential to increase client utility with the larger number of uncertainty intervals. This also can be explained, considering that our evaluation algorithm (see Section 3.3.3.3) chooses an extreme $\beta^c_{i,k}$ (i.e. the smallest or the largest) and as closer a triangle's side to its top position as $\beta^c_{i,k}$ more likely can be increased or decreased rapidly. Thus, the large number of segments, on which the interval of uncertainty is divided, may provide a client with the more significant decrease of $\beta^c_{i,k}$ in the case of resource decreasing. As a result, it leads to the larger number of successful negotiations and an improvement in utility where the risk of resource exhaustion is higher.

Moreover, our strategies 'Fuzzy&Var_2st', 'Fuzzy&Var_5st' and 'Fuzzy&Var_10st' outperform a strategy 'FullKnow' for all probabilities of tendency. In Figure 3.14, which shows a distribution of resources among tasks for the high speed of resource dynamism,

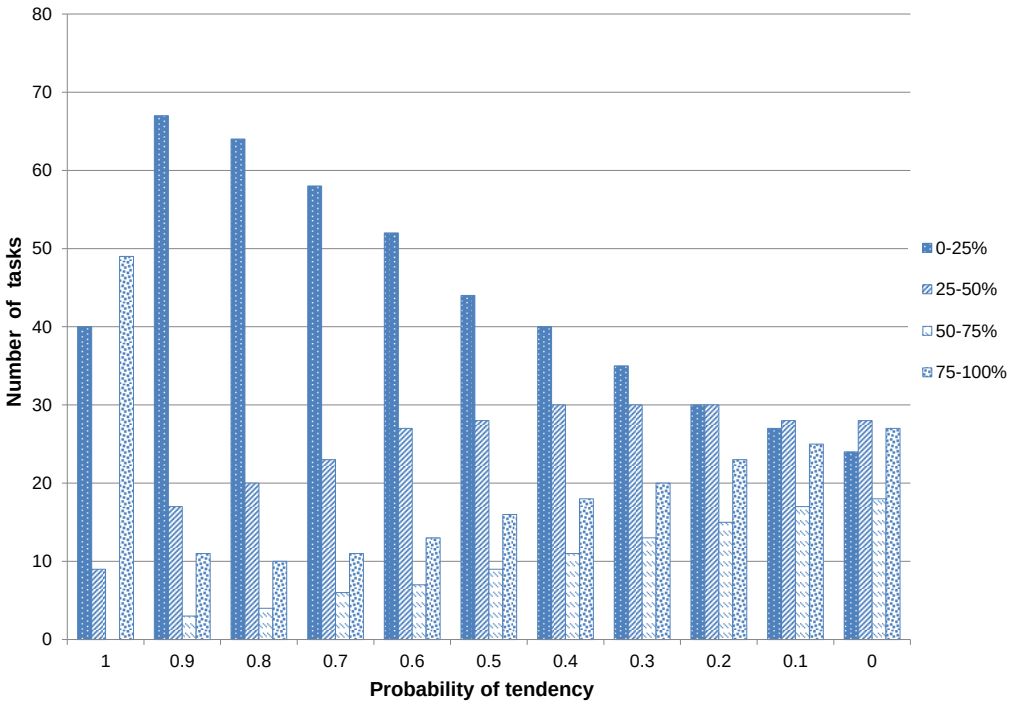



(a) Client Utility

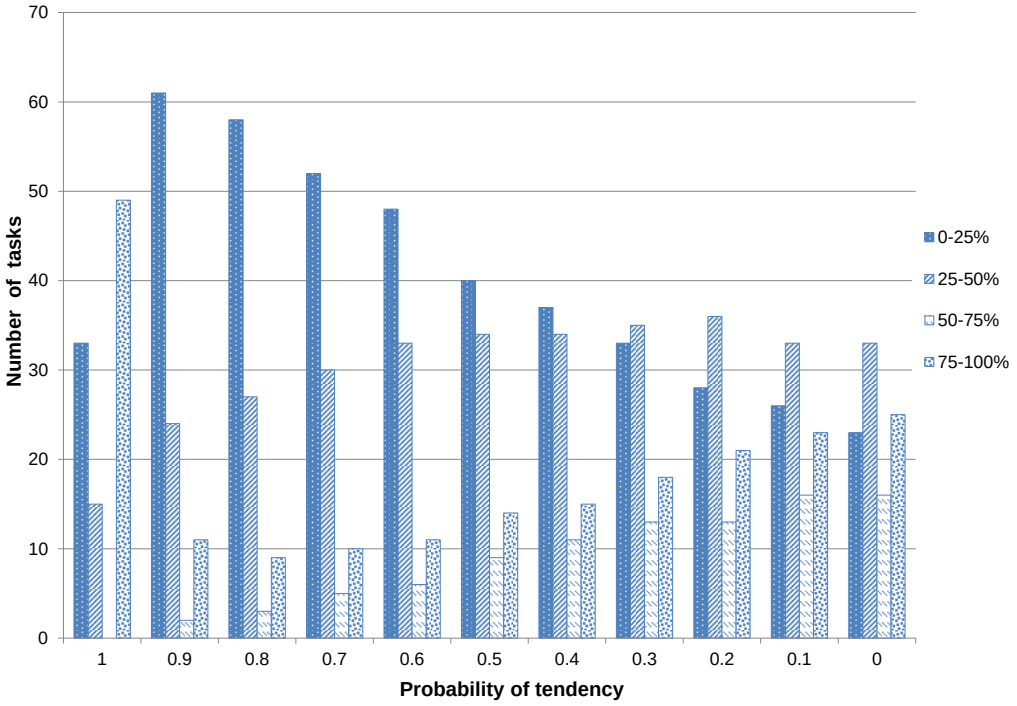



(b) Number of successful negotiations

FIGURE 3.13: Client utilities for the high speed resource dynamism

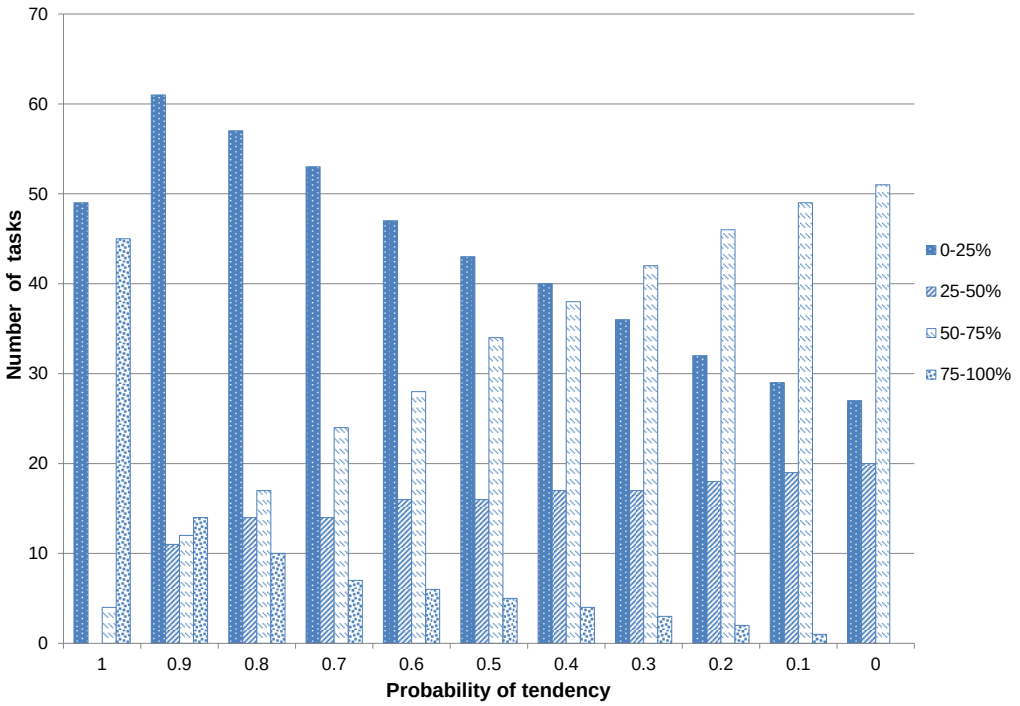



(a) Resource distribution for 'Fuzzy&Var_2st'

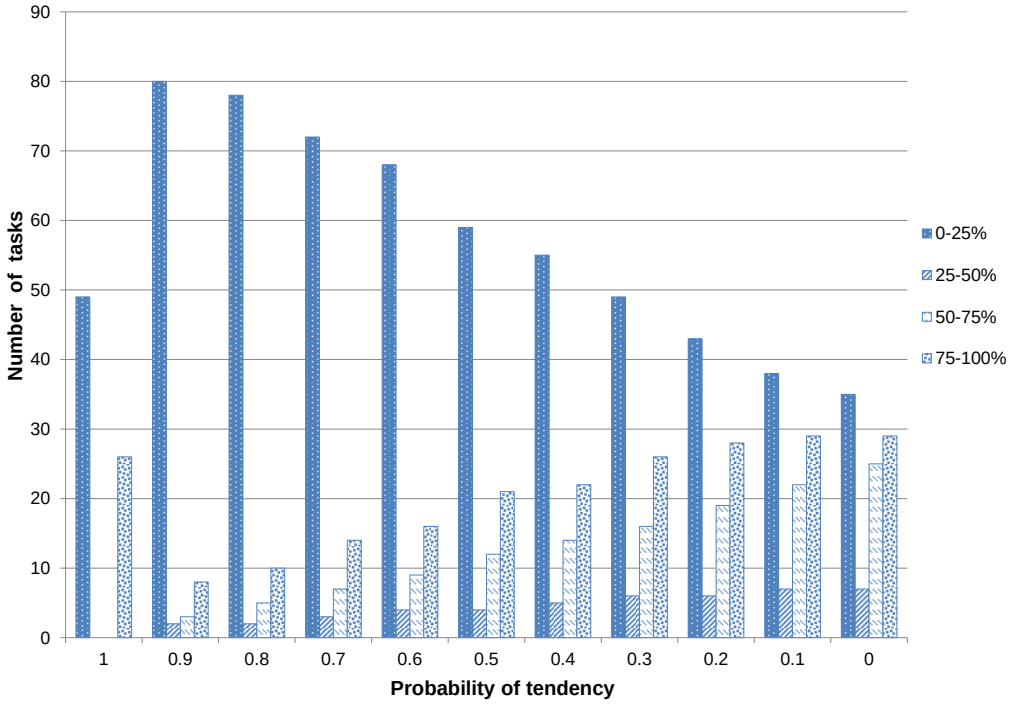



(b) Resource distribution for 'Fuzzy&Var_5st'

FIGURE 3.14: Resource distribution for the high speed resource dynamism in the cases 'Fuzzy&Var_2st' and 'Fuzzy&Var_5st'




(a) Resource distribution for 'Fuzzy&noVar'




(b) Resource distribution for 'FullKnow'

FIGURE 3.15: Resource distribution for the high speed resource dynamism in the cases 'Fuzzy&noVar' and 'FullKnow'

the tasks for the strategy 'FullKnow' are mostly concentrated in the range group $0-25\%$ for the majority of the probabilities. However, the number of tasks, which belongs to the range groups $50-75\%$ and $75-100\%$ (see Figure 3.15(b)), gradually increases when the risk of resource exhaustion decreases towards the less tendentious Grid. It has to be noted that the strategy 'Fuzzy&noVar' increases the number of tasks in the range group $50-75\%$ and decreases the number of tasks in the range group $75-100\%$ for the lower probabilities of tendency (see Figure 3.15(a)), while all other strategies mostly intends to increase the number of tasks in the range group $75-100\%$. Therefore, the strategy 'Fuzzy&noVar' shows the lower utilities for the less tendentious Grid. Our strategy 'Fuzzy&Var_5st' (see Figure 3.14(b)) gains the lower utilities for a client than 'Fuzzy&noVar' in the interval $[0.9, 0.6]$ mostly because it has a significantly less number of tasks in the range group $50-75\%$ but a larger number of tasks is in the range group $25-50\%$. This also can be explained by the unpredicted increases or decreases in resource availability. For example, if the resource availability tendentiously decreases, a client becomes more and more generous. When resource availability suddenly increases, a client with our strategy 'Fuzzy&Var_2st', 'Fuzzy&Var_5st' (see Figures 3.14(a), 3.14(b)) continues to think that the main direction is negative and loses in utility, obtaining the smaller amount of resource.

Finally, our strategy outperforms all other strategies for the low speed resource dynamism and knows the larger number of successful negotiations with the larger number of steps for the high speed resource dynamism. It also outperforms other strategies for the lower tendencies for any simulated speed of resource dynamism. The larger number of steps also demonstrates a potential to improve client utility for the higher tendencies for the high speed resource dynamism.

## 3.5 Conclusions

This chapter has described our adaptive negotiation strategy for a client whose ultimate goal is to obtain the acceptable resource amounts for as many tasks as possible and these resource amounts should preferably be closer to the tasks' respective maximum resource requirements. A client is assumed to be self-interested and, therefore, it starts negotiation with a reasonably greedy behaviour, while the GRA is indifferent in respect of all clients at the beginning of negotiation. The proposals of a client and the GRA depend on the resource availability in the Grid, where the GRA has a full knowledge

about resources, while a client has to obtain relatively accurate information about resource availability through negotiation with the GRA. We also assume that the GRA changes its negotiation parameters, which can be inferred by a client, only in response to the changes in resource availability. Consequently, a client adapts its tactics to the tendencies in resource availability changes, inferred through the GRA's proposals during negotiation. This strategy allows a client to respond to any changes in resource availability, starting from the early negotiation rounds without a prior knowledge about the resource availability. Here, a client estimates the risk of resource exhaustion, which causes a failure of negotiation, and tries to reach an agreement before this happens. If there is no such risk, then a client tries to obtain the larger amount of resources, negotiating for the longer times.

In our evaluation, we tested our strategy for the whole spectrum of tendencies in resource availability changes in the Grid, comparing the resulting utilities for the different input settings and other strategies. The two major settings show the speed of the changes of the GRA's reservation value (consequently, the resource availability), where the low speed denotes smaller changes per round and high speed denotes larger changes per round. Our simulation results show that our negotiation strategy improves the client's utility for all probabilities of tendency in the case of the low speed changes, compared to all other modelled strategies. It even outperforms the strategy with a full knowledge about its opponent, which does not take into consideration the risk of resource exhaustion. Our strategy also increases the number of successful negotiations for the low and high speeds of resource dynamism, while it shows a slightly smaller utilities than the strategy which does not apply a variation of uncertainty intervals for the larger probabilities of tendency in the case of high speed of resource dynamism, caused by the less predictable large deviations of the GRA's reservation value.

# Chapter 4

# Adaptive Negotiation for Continuous Tasks

## 4.1 Introduction

Nowadays, much research has been conducted into processing data streams [6, 29–32] from sensors, which represents a motivating scenario for our work.[1] Some of these data streams have to be processed continuously for long periods of time e.g., data from sensors which monitor the level of pollution or the possibility of earthquake. This data can be used to control the monitored parameters e.g., the level of pollution. Therefore, it is desirable for these data streams to be processed in real-time and this processing has to avoid interruptions of such length when the unpredictable changes may happen in the environment. However, a comparably short interruption of data processing should not cause a disaster, as discussed in Chapter 2. We assume that each task can be re-launched correctly as soon as an acceptable resource has been allocated without a regard to its past data loss, i.e. a task does not use past data to process current real-time data. Nevertheless, a task's interruption means that some data has not been processed in time, which negatively affects the client utility. For example, the sensors may monitor temperature in a greenhouse (we assume a greenhouse as an element of a smart city (see Chapter 1)) and this data can be used to control this climate parameter in real-time. Assume that the data processing from sensors is interrupted due to the lack of computational resources or other causes and, as a result control of the temperature

[1] The contribution, presented in this chapter, has been published in the paper [27].

in a greenhouse is terminated due to the absence of data. In this situation, the plants are more likely to survive for a short period of time, because temperature inside the greenhouse would fall relatively slowly, depending on the outside temperature and other environmental conditions. We can estimate when an interruption of processing these data streams will start to cause damage to the plants. At the point in time when a decrease of temperature starts significantly affecting well-being of the plants, an owner of the greenhouse starts noticeably losing in utility. However, this process is gradual (but the speed of utility decrease may be different for tasks) in real-life scenarios and the decrease in the owner's utility is also a gradual continuous process.

We assume that a continuous task, as discussed above, is appropriate to execute in a Grid, because it potentially requires more computational, storage and other resources than provided by any supercomputer or cluster. However, a Grid can be an open environment which is accessible for any user who wishes to run a program and, therefore, the resource availability may vary over time and be scarce at some points in time. Moreover, some resources may leave or join a Grid in future. Note that the resource availability often changes periodically over time. For example, the resources might be more busy during daytime when more clients run their programs, but they can be less busy at night (see Section 2.4.2). Considering a continuous task might be run for weeks, months or years, a Grid is unlikely to be willing or able to allocate resources for this task for all of the required long-term duration, because of the high dynamism in its resource availability and / or its policy. In this case, the task will experience *planned* interruption after some period of execution.

In our work, resources are allocated to tasks after an agreement is reached between a client and the Grid Resource Allocator (GRA). Therefore, our goal is to improve the client's negotiation strategy to reach an agreement faster and with a relatively higher amount of resources allocated. In Chapter 3, we developed a client's negotiation strategy which adapts to changes in resource availability, applying a fuzzy control mechanism, where a larger resource amount may denote more CPU cores, more disk space or bandwidth. In this chapter, the resource amount is an *allocation* period of time during which a task is running until the next planned interruption. An *interruption* period is a time slot during which a task is not running and the client is negotiating with the GRA over resources. The whole long-term duration of task execution (e.g. one year) is called an *execution* period of time. In our work, we also consider that a task can be interrupted not only due to the inability of the Grid to schedule its execution far in advance, but also due to resource failure of any kind (e.g. a hard

drive failure, a connection failure). If a task is interrupted before reaching its planned interruption, then this interruption is called *unexpected* and it means that the client may suddenly find itself in a situation when there are no available resources and the expected utility for the allocation period is not reached. In the case of the planned interruption, a client is able to estimate approximately the expected level of resource availability, based on the assumption that resource availability changes periodically. In comparison, an unexpected interruption does not allow a client to predict beforehand the level of resource availability at the time of negotiation.

This chapter describes a formal model for continuous long-term tasks in Section 4.2, a proposed negotiation strategy for a client in Section 4.3 and results in Section 4.4. Sections 4.2 and 4.3 include tables of notations, which are introduced in these sections (see Tables 4.1 and 4.2).

## 4.2 Formal Model

In Chapter 3, we considered a situation where a client and a GRA may negotiate over an abstract Grid resource e.g., the number of CPU cores. In the model below, the client negotiates with the GRA for a resource, and this resource is defined as a time duration of utilisation of a particular physical Grid resource. We focus on the allocation of time rather than a physical resource in this model because the problem we intend to solve is the continuity of task execution over time. Therefore, time becomes a natural objective for negotiation. We also assume in this model that a task can be interrupted during its execution and, therefore, it has to be re-run repeatedly until it completes its execution period of time. A client has to negotiate with the GRA for a new time slot every time, when a task has been interrupted for any reason, i.e. a planned or unplanned interruption (e.g. a resource failure). This formal model focuses on a formalisation of the repeated allocation of time slots after respective interruptions for a task over time, while our previous model only considered a single execution of a task until its completion or interruption.

The structure of this section is as follows. Section 4.2.1 extends the number of task's attributes and modifies the specification of a task from Chapter 3, assuming a task has to be re-run repeatedly and a negotiated resource is a duration of time. Then, Section

4.2.2 modifies the model of resource availability fluctuation in a Grid, considering periodicity of resource availability. Finally, Section 4.2.3 describes a client's utility function for the case of the repeated task allocations over time.

### 4.2.1 Task Description

A continuous task denotes a task for which it is desirable to avoid interruptions for the whole long-term duration of execution. Assume that the start time of a task $i$ is $t^0$ and the deadline[2] to complete an execution of this task is $t_{dl}^{exec}$. A task $i$ is submitted to a Grid with other client tasks in one job. As in our previous model, we assume that all tasks are independent in terms of execution, but they have some common attributes. We consider that each task has the same start $t^0$ and end $t_{dl}^{exec}$ times. We also assume that the initial negotiation for resources denotes the start time $t^0$ of an execution period for each task as depicted in Figure 4.1.

*Definition* 4.1. An execution period $\tau_{dl}^{exec} > 0$ for a continuous task $i \in \mathbb{N}$ is a period of time within which a task is aimed to be run continuously which starts at time $t^0 \in \mathbb{R}$ and ends at time $t_{dl}^{exec} \in \mathbb{R}$, including the periods of time when a task is not running.

*Definition* 4.2. An allocation period $\tau_i^{all} > 0$ for a continuous task $i \in \mathbb{N}$ is a period of time within an execution period $\tau_{dl}^{exec} > 0$ when the task is running.

*Definition* 4.3. An interruption period $\tau_i^{int} > 0$ for a continuous task $i \in \mathbb{N}$ is a period of time within an execution period $\tau_{dl}^{exec} > 0$ when the task is not running.

Figure 4.1 schematically shows a process of execution of task $i$, which is modelled in our work. As shown in this figure, the allocation periods always follow specific interruption periods of time during which a client negotiates with the GRA.

*Notation* 4.1. A pair of adjoined interruption-allocation periods $\left(\tau_i^{int}, \tau_i^{all}\right)_l$, where an interruption period precedes an allocation one, has its counter $l = 1, 2, 3, ...$ within an execution period $\tau_{dl}^{exec}$. The allocation period in this pair starts at time $t_{i,l}^{str} \in \mathbb{R}$ and finishes at time $t_{i,l}^{end} \in \mathbb{R}$, where $i, l \in \mathbb{N}$.

Here, we consider a continuous task which has to be executed for as long time as possible without interruptions, and which is sensitive in respect of those interruptions. Consequently, we focus mostly on the cases when the interruption periods are intended to be significantly shorter than the allocation periods based on our motivating example

[2] In our work, the moments of time are indicated as $t$ and the periods of time are indicated as $\tau$.

with a greenhouse, i.e. $\tau_{i,l}^{int} << \tau_{i,l}^{all}$. For example, an interruption may take seconds or minutes, while an allocation period can be up to days, considering long-term task execution.

The length of each single interruption period affects the utility which a client gains for the allocation period obtained during negotiation. That is, the longer interruption period, the less utility will be obtained for the allocation period. This influence of a single interruption on the client's outcome is described by a pair of task attributes $(\tau_{int}^{max}, \epsilon_{int})$ (see Equation (4.7)), which denotes how fast a possible increment of the client utility for the next allocation period decreases towards zero while within an interruption period. That is,

*Definition* 4.4. An attribute $\tau_{int}^{max} > 0$ for task $i \in \mathbb{N}$ (i.e. $\tau_{int[i]}^{max}$) is the duration of a single interruption wherein the increment of the client utility in respect of the obtained allocation period decreases to half its possible value, if the allocation period is obtained at once after a task has been interrupted.

*Notation* 4.2. An attribute $\epsilon_{int} > 0$ for task $i \in \mathbb{N}$ (i.e. $\epsilon_{int[i]}$) determines the speed of decrease in the corresponding increment of the client utility in respect of the obtained allocation period, caused by the length of the interruption period, and this speed increases in the approximate proximity of the end of time period $\tau_{int[i]}^{max}$.

We also assume that not only the duration of a single interruption affects the client utility for a task $i$, but also the total duration of interruptions $\tau_{i,l}^{tot}$ prior to each allocation period $\tau_{i,l}^{all}$. For example, in Figure 4.1 the total duration of interruptions $\tau_{1,2}^{tot}$ prior to the allocation period $\tau_{1,2}^{all}$ is the sum of $\tau_{1,1}^{int}$ and $\tau_{1,2}^{int}$. Consequently, the total duration of interruptions $\tau_{1,3}^{tot}$ prior to $\tau_{1,3}^{all}$ is the sum of $\tau_{1,1}^{int}$, $\tau_{1,2}^{int}$ and $\tau_{1,3}^{int}$. This duration cumulates during task execution, affecting every following allocation period.

*Definition* 4.5. A duration of time $\tau_{i,l}^{tot} = \sum_{k=1}^{l} \tau_{i,k}^{int}$ for task $i \in \mathbb{N}$ is the total duration of interruptions over the execution period $\tau_{dl}^{exec}$ prior to the allocation period $\tau_{i,l}^{all}$, where $l \in \mathbb{N}$.

The longer this total interruption, the less successful an overall execution of the task. In this way, a client has to aim to decrease the length of this total interruption $\tau_{i,l}^{tot}$ to be much smaller than the specified execution period $\tau_{dl}^{exec}$, i.e. $\tau_{i,l}^{tot} << \tau_{dl}^{exec}$. The influence of the total duration of interruptions on the client's outcome is described by a pair of task attributes $(\tau_{tot}^{max}, \epsilon_{tot})$ (see Equation (4.8)), which describes how fast a possible increment of the client utility for the next allocation period drops towards zero while the total duration of interruptions increases.

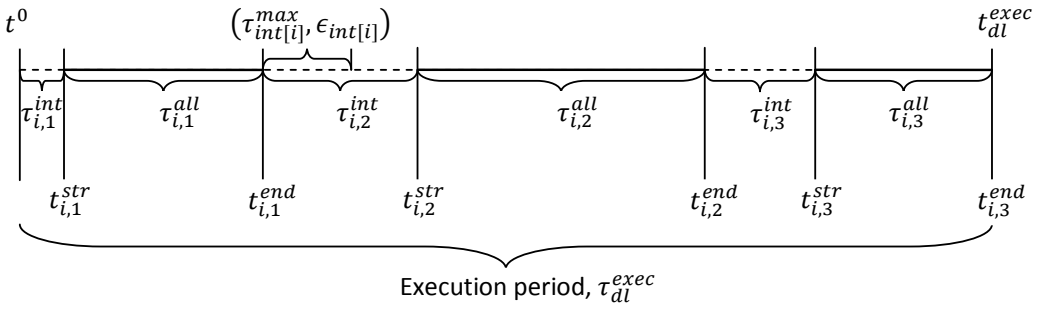



FIGURE 4.1: The process of continuous task execution

*Definition* 4.6. An attribute $\tau^{max}_{tot} > 0$ for task $i \in \mathbb{N}$ (i.e. $\tau^{max}_{tot[i]}$) is the total duration of interruptions within an execution period $\tau^{exec}_{dl}$ wherein every increment of the client utility in respect of the allocation periods decreases to half its possible value, if a total interruption is not considered (this decrease is independent from a decrease in the client utility caused by a single interruption which is discussed above).

*Notation* 4.3. An attribute $\epsilon_{tot} > 0$ for task $i \in \mathbb{N}$ (i.e. $\epsilon_{tot[i]}$) determines the speed of decrease in the increment of the client utility in respect of the obtained allocation period, caused by the length of the total interruption period prior to this allocation period, and this speed increases in the approximate proximity of the end of time period $\tau^{max}_{tot[i]}$.

The use of these task attributes in the client utility is explained in detail in Section 4.2.3.

In our work, each task has attributes which are available only to the client, but not to the GRA. Therefore, this information is considered to be *private*. The new attributes, compared to Chapter 3, include $t^{exec}_{dl}$, $\epsilon_{int[i]}$, $\tau^{max}_{int[i]}$, $\epsilon_{tot[i]}$ and $\tau^{max}_{tot[i]}$. The start time $t^0$ of execution of task $i$ is specified automatically as soon as a client has submitted its resource request to the GRA in respect of its tasks. It has to be noted that in our current model, the values of the corresponding task attributes are identical for all tasks as multiple tasks are mostly considered for statistics. However, our next Chapter 5 considers inter-dependent tasks which might have different values of these attributes for the different tasks. Here, the attributes for task $i$ are presented in the following formula.

$$Task_i = \left(\tau^{max}_{int[i]}, \epsilon_{int[i]}, \tau^{max}_{tot[i]}, \epsilon_{tot[i]}, t^{exec}_{dl}, t^{c}_{dl}\right), \tag{4.1}$$

where $t_{dl}^{c}$ describes a nominal deadline for a single negotiation in this model.

The specification of task $i$ is considered to be available to the GRA, but not to the other clients in a Grid. It describes how task $i$ has to be run on a Grid, stating specific resource requirements. In this chapter, this specification indicates an maximum duration of resource utilisation $\tau_{i,l}^{opt}$ and a minimum acceptable duration of resource utilisation $\tau_{i,l}^{min}$ for task $i$, while inside the interruption period $\tau_{i,l}^{int}$. Here, we assume that it is reasonable for a Grid to inform all clients about the possible duration of task execution that it may provide to reduce the duration of negotiation. Therefore, a Grid announces up to what duration of time it may provide its resources for a task, based on its policy or the current demand on resources. For example, this duration can be a few days. This also complies with our assumption that a Grid cannot always allocate resources for the tasks for the whole duration they need to run.

In this way, a client's *maximum* duration of task execution $\tau_{i,l}^{max}$ that it asks for is the possible duration specified by a Grid and it changes in our model each interruption period $\tau_{i,l}^{int}$. We believe that it is realistic to assume that the GRA's policy or resource availability may significantly change only after a relatively long period of time such as an allocation period, but not inside one interruption period which is considerably shorter. A client's *minimum* duration $\tau_{i,l}^{min}$ depends on the maximum one and if the maximum duration of time becomes longer in the next interruption period, then the minimum duration also has to become longer to avoid losing in utility and vice versa. The definitions of the maximum and minimum durations of task execution are described as follows.

*Definition* 4.7. The maximum duration of execution $\tau_{i,l}^{max} > 0$, $\epsilon_{tot} \in \mathbb{R}$ for task $i$ is the maximum duration of time a client may ask from the GRA during a single negotiation within a particular interruption period $\tau_{i,l}^{int}$, and this duration is restricted by the GRA based on the demand on resources or other considerations, where $i, l \in \mathbb{N}$.

*Definition* 4.8. The minimum duration of execution $\tau_{i,l}^{min} > 0$, $\epsilon_{tot} \in \mathbb{R}$ for task $i$ denotes the minimum duration of time a client is willing to accept from the GRA during a single negotiation within a particular interruption period $\tau_{i,l}^{int}$, and this duration proportionally depends on the maximum duration of execution $\tau_{i,l}^{max}$, where $i, l \in \mathbb{N}$.

Considering the discussion above, the task's specification $Spec_{i,l}$ consists of a description of time resources required to run a particular task $i$. Then, the task associated

with this specification has to be run continuously for a long period of time.

$$Spec_{i,l} = \left(i, \tau_{i,l}^{min}, \tau_{i,l}^{max}\right). \tag{4.2}$$

In our work, a client submits multiple tasks to a Grid in one job for the initial negotiation (when no tasks have been allocated any resources) as described in Definition 3.1. As long as these tasks are executed independently, the negotiations over them are also considered to be independent. Then, a client can send a message to the GRA which contains an updated task description $Spec_{i,l}$ ($\tau_{i,l}^{max}$ and $\tau_{i,l}^{min}$ may change for the different $\tau_{i,l}^{int}$) every interruption period $\tau_{i,l}^{int}$ in order to initiate a negotiation for task $i$ (instead of submitting the whole job), while another message with an offer $Pr_{i,j,t}$ is sent every round $j$ at time $t$ during this negotiation process. A proposal from any negotiator in round $j$ at time $t$ for task $i$ contains the identifier $i$ of a task and a proposed allocation period $\hat{\tau}_{i,j}^{all}(t)$ as described in Formula (4.3). Here, $t$ denotes the current moment of time of task execution which counts starting from $t^0$ and affects the client's proposal as a result of the pseudo-periodicity in resource availability changes and client's considerations in respect of the single and total interruptions.

$$Pr_{i,j,t} = \left(i, \hat{\tau}_{i,j}^{all}(t)\right). \tag{4.3}$$

### 4.2.2 Resource Dynamism

As in our previous chapter, a client estimates the changes in resource availability in a Grid by taking into account the GRA's proposals. The GRA's proposals significantly depend on how its reservation value changes over negotiation rounds, according to the time-dependent strategy proposed by Faratin et al. [17]. Therefore, in order to evaluate how effectively a client negotiates with the GRA, we model the change of the GRA's reservation value over time and this model is described in this section. As we discussed previously in this chapter, we assume that the amount of available resources changes pseudo-periodically over time, which means that the representation of the GRA's reservation value has to contain a periodic function, because its value changes depending on the amount of available resources in a Grid. Moreover, in our previous work, we state that the reservation value of the GRA has to be smaller or equal to the client's maximum value $\tau_{i,l}^{max}$ as long as the GRA does not intend to provide a client with more resources (in this case the duration of time) than it needs. We also assume that the GRA's reservation value should not be smaller than the client's minimum

duration of task execution $\tau_{i,l}^{min}$, because there is no sense in making a proposal that cannot be accepted by the client.

Considering the discussed bounds on the change in the GRA's reservation value, we can model this by deducting $Value_{i,l}(t)$ from $\tau_{i,l}^{max}$, where $Value_{i,l}(t)$ is a periodically changing function over time $t$ that does not exceed $\tau_{i,l}^{max} - \tau_{i,l}^{min}$ unless the resource availability is around its periodic minimum. That is, $Value_{i,l}(t)$ is a proportion of negotiation interval $\tau_{i,l}^{max} - \tau_{i,l}^{min}$ which is deducted from $\tau_{i,l}^{max}$ to simulate a change in the GRA's reservation value $G_{i,j,l}^{max}(t)$. Here, we consider that $G_{i,j,l}^{max}(t)$ has to be smaller or equal to $\tau_{i,l}^{max}$ and larger or equal to $\tau_{i,l}^{min}$ for negotiation to proceed, and it might be smaller than $\tau_{i,l}^{min}$ in the cases when resources are less available which denotes resource exhaustion for a client. Therefore, the reservation value of the GRA in every negotiation round can be calculated as $\tau_{i,l}^{max} - Value_{i,l}(t)$, where $Value_{i,l}(t)$ is described in the following equation:

$$Value_{i,l}(t) = K_1 \times \left(\tau_{i,l}^{max} - \tau_{i,l}^{min}\right) \times \left(1 - K_2 \times \sin\left(\frac{2\pi}{T_{res}} \times t\right)\right), \tag{4.4}$$

where $0 < K_1 \leq 1$ and $0 < K_2 < 1$ are experimentally chosen coefficients, which determine the amplitude of periodic fluctuations of the GRA's reservation value and the value of $K_1 \times \left(1 - K_2 \sin\left(\frac{2\pi t}{T_{res}}\right)\right)$ determines a proportion of the negotiation interval $\tau_{i,l}^{max} - \tau_{i,l}^{min}$. Here, the coefficients $K_1$ and $K_2$ regulate the minimum and maximum of the GRA's reservation value, their difference and the time ranges of resource scarcity. That is, those coefficients aim to simulate various Grid environments. The period of resource availability changes is indicated by $T_{res}$.

However, in a real life scenario, the reservation value of the GRA is unlikely to change according to some strict periodic function, because the change in resource availability is usually pseudo-deterministic [11]. Therefore, the GRA's reservation value will also have stochastic deviations over time. Hence, a random component is necessary to simulate the GRA's reservation value, but we also intend to control this random component in some way to be able to analyse the results. In this way, a random component can be presented as in Formula (4.5).

$$Random_{i,j,l} = \left((K_3/100\%) \times \tau_{i,l}^{max}\right) \times rand_j, \tag{4.5}$$

where $K_3 > 0$ is the percentage of $\tau_{i,l}^{max}$, which determines the standard deviation of the GRA's reservation value in respect of $\tau_{i,l}^{max} - Value_{i,l}(t)$ i.e. the larger $K_3$, the larger the deviations that may occur; $rand_j$ denotes a randomly generated number from the

normal distribution. Considering the discussion above, we specify that $G^{max}_{i,j.l}(t)$ in round $j$ at time $t$ is generated by the GRA using the following equation:

$$G^{max}_{i,j,l}(t) = \tau^{max}_{i,l} - Value_{i,l}(t) + Random_{i,j,l}. \tag{4.6}$$

In Formula (4.6), a random component potentially may increase the value of $G^{max}_{i,j,l}(t)$ above the value of $\tau^{max}_{i,l}$. Therefore, in the case when $G^{max}_{i,j,l}(t) > \tau^{max}_{i,l}$, the value of $G^{max}_{i,j,l}$ is algorithmically equalised to $\tau^{max}_{i,l}$.

### 4.2.3 Client Utility

A client's utility reflects how the client expects its tasks should be executed on a Grid. In our work, utility has to take into account the duration of a single interruption as well as the total duration of interruptions within an execution period, and should decrease gradually the longer these interruptions are. The allocation period, compared to the interruption one, improves the client utility by increasing it gradually the longer this period is. The utility gained by a client for each allocation period is affected negatively to a varying degree by two factors:

1. The duration of interruption period of time prior to a particular allocation period of time, which shows how effectively a client negotiated for this allocation period;
2. The total duration of interruption periods prior to a particular allocation period, which shows how effectively a client generally negotiates for allocation time.

First, we introduce our *effectiveness* function $E(t)$ which denotes the success of the execution of task $i$ over time, assigning a value from the range $[0, 1]$ for each time unit $t$ (see Figure 4.2). In Figure 4.2, the dashed line $AB$ shows a would-be effectiveness function, if there were no interruptions during the whole execution period $\tau^{exec}_{dl}$. An actual effectiveness function in the case of interruptions is presented as a polyline in this figure, which linearly increases during allocation periods and does not change during interruption periods. Each interruption period $\tau^{int}_{i,l}$ affects the following allocation period $\tau^{all}_{i,l}$ by reducing the worth of the obtained resources. That is, the overall effectiveness of task execution decreases after an interruption period, including total interruption periods, and it starts from the smaller value of effectiveness, while increasing during the following allocation period. The effectiveness function increases linearly when the task

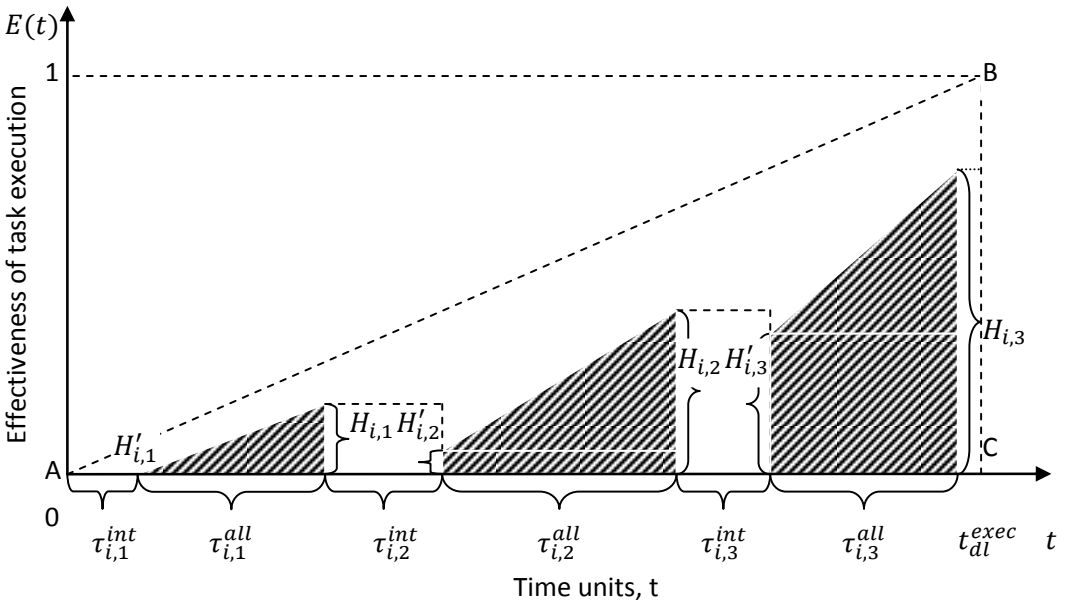



FIGURE 4.2: The effectiveness of task execution over time

is running and it does not change when the task is interrupted. However, the length of interruption affects the value of effectiveness from which the effectiveness function starts at the beginning of the next allocation period and the angle of this effectiveness function. The shaded areas in the figure contribute into the client's utility gained at the end of task execution, which is discussed in this section below.

Formally, the influence of the interruptions on the effectiveness of task execution can be represented with the damping functions $I(\cdot)$ and $T(\cdot)$. One of these functions $I\left(\tau^{int}_{i,l}\right)$ (see Figure 4.3) maps the duration of a single interruption $\tau^{int}_{i,l}$ to the value from the interval $[0,1]$ for each allocation period $\tau^{all}_{i,l}$. Another function $T\left(\tau^{tot}_{i,l}\right)$ maps the total length of the interruptions $\tau^{tot}_{i,l}$ which occurred before a particular allocation period $\tau^{all}_{i,l}$ to the value from the interval $[0,1]$ for this allocation period. Both damping functions have the *inflection points* $\tau^{max}_{int[i]}$ (see Definition 4.4) and $\tau^{max}_{tot[i]}$ (see Definition 4.6) for $I\left(\tau^{int}_{i,l}\right)$ and $T\left(\tau^{tot}_{i,l}\right)$ respectively. These functions also have such parameters as $\epsilon_{int[i]}$ (see Definition 4.2) and $\epsilon_{tot[i]}$ (see Definition 4.3) which define the speed of decreasing of the damping functions $I\left(\tau^{int}_{i,l}\right)$ and $T\left(\tau^{tot}_{i,l}\right)$ before and after the inflection points.

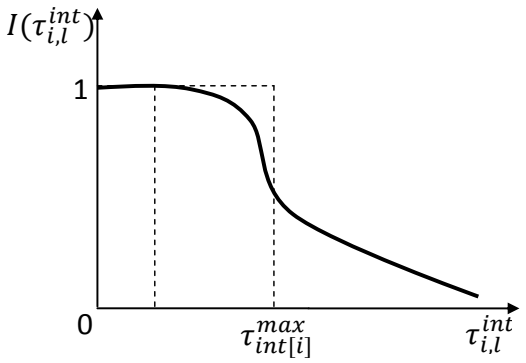



FIGURE 4.3: Client evaluation of the interruption duration

The functions $I\left(\tau_{i,l}^{int}\right)$ and $T\left(\tau_{i,l}^{tot}\right)$ are presented in Formulas (4.7) and (4.8).

$$I\left(\tau_{i,l}^{int}\right) = \frac{1}{e^{\left(\tau_{i,l}^{int} - \tau_{int[i]}^{max}\right)/\epsilon_{int[i]}} + 1}, \tag{4.7}$$

$$T\left(\tau_{i,l}^{tot}\right) = \frac{1}{e^{\left(\tau_{i,l}^{tot} - \tau_{tot[i]}^{max}\right)/\epsilon_{tot[i]}} + 1}, \tag{4.8}$$

The larger the value of $\epsilon_{int[i]}$ ($\epsilon_{tot[i]}$), the sooner a damping function starts noticeably falling towards zero from the beginning of the interruption period (from the beginning of the first interruption period). The smaller the value of $\epsilon_{int[i]}$ ($\epsilon_{tot[i]}$), the longer a damping function does not fall noticeably towards zero and the sharper it starts decreasing closer to the inflection point. These two different behaviours of the damping functions depend on the nature of a client task e.g., an example with the greenhouse. We also assume that the effectiveness should not be equal to zero (but it can be close to zero) after some time, because in a real life scenario the client system is unlikely to become absolutely non-recoverable after a predefined period of time (e.g. there is always a possibility that some plants may survive even when temperature falls below an acceptable threshold). Moreover, if we assumed that the effectiveness was equal to zero after some time, this would mean we have no reason to continue task execution as long as the client system has already received a non-recoverable damage.

The linear function $E\left(t\right)$ which describes an increase of the effectiveness during the allocation period changes in the range from the level of effectiveness achieved by the task before interruption $E\left(t_{i,l-1}^{end}\right)$ at the moment of time $t_{i,l-1}^{end}$ (it does not change

during the interruption period), multiplied by the values of functions $I\left(\tau_{i,l}^{int}\right)$ and $T\left(\tau_{i,l}^{tot}\right)$, towards the largest level of effectiveness 1 at the moment of time $t_{dl}^{exec}$ on the time interval $\tau_{i,l}^{all}$. We assume that when a task obtained an allocation time, it is launched immediately. Consequently, the effectiveness function at the current moment of time $t$ is presented in the following recursive equation:

$$E\left(t\right) = \begin{cases} \left[\frac{\left(1 - E\left(t_{i,l-1}^{end}\right)\right) \times t + E\left(t_{i,l-1}^{end}\right) \times t_{dl}^{exec} - t_{i,l}^{str}}{t_{dl}^{exec} - t_{i,l}^{str}}\right] I\left(\tau_{i,l}^{int}\right) T\left(\tau_{i,l}^{tot}\right), & if\ \tau_{i,l}^{all} \neq 0 \\ E\left(t_{i,l-1}^{end}\right) & , if\ \tau_{i,l}^{all} = 0, \end{cases} \tag{4.9}$$

where $\tau_{i,l}^{all} \neq 0$ means that the task is running, while $\tau_{i,l}^{all} = 0$ denotes that the task is not running.

The effectiveness function shows the success of task execution as a process, while we need a function that produces a final single value to judge this process. This final value is *utility* that the client gains after the task is executed for a period of time $\tau_{dl}^{exec}$. If a client task obtains an allocation time at the first round of initial negotiation $t^0$, then the effectiveness function will change from zero to one linearly (see Figure 4.2) on the time interval $\left[t^0, t_{dl}^{exec}\right]$. In this best scenario, a client obtains the largest possible utility, which we estimate as the square of an area under the effectiveness function for each task. This largest utility is considered to be the same for all tasks in one job, because their execution periods start ($t^0$) and end ($t_{dl}^{exec}$) at the same time. Then, the largest possible square $S_{max}$ is shown in the following equation as the square of a triangle 'ABC':

$$S_{max} = \frac{1}{2} \times 1 \times \tau_{dl}^{exec}. \tag{4.10}$$

Comparably, the smallest square is equal to zero, which means that the task was not executed at all during the execution period $\tau_{dl}^{exec}$. Only that part of the square $S_{max}$ when a task is running contributes to the client utility. It is calculated as the sum of the filled trapeziums as depicted in Figure 4.2, where $H_{i,l}$ or $H_{i,l}^{'}$ are the sides and $\tau_{i,l}^{all}$ is the height of each trapezium for task $i$. Considering $t_{i,l}^{str}$ and $t_{i,l}^{end}$ for allocation period $\tau_{i,l}^{all}$, then $H_{i,l}^{'} = E\left(t_{i,l}^{str}\right)$ and $H_{i,l} = E\left(t_{i,l}^{end}\right)$. Consequently, the sum of trapeziums $S_{sum}^{i}$ and the client utility $U_{Cl}^{i}$ for task $i$ is presented in the following equations:

$$S_{sum}^{i} = \frac{1}{2} \sum_{l=1}^{L_i} \left(H_{i,l}^{'} + H_{i,l}\right) \times \tau_{i,l}^{all}, \tag{4.11}$$

where $L_i$ is the total number of the allocation periods within the execution period $\tau_{dl}^{exec}$ for task $i$.

$$U_{Cl}^{i} = \frac{S_{sum}^{i}}{S_{max}}, \tag{4.12}$$

where $U_{Cl}^{i} \in [0, 1]$. The average client utility $U_{Cl}^{aver}$ for all tasks $N$ in one job can be estimated as in the following equation:

$$U_{Cl}^{aver} = \frac{1}{N} \sum_{i=1}^{N} U_{Cl}^{i}. \tag{4.13}$$

TABLE 4.1: List of notation for Section 4.2

| Symbol | Notation |
|---|---|
| $t^0$ | The start time of the initial resource negotiation (before any task in one job has been allocated resources for any duration of time) for each task $i \in \mathbb{N}$ in one job, where $t^0 \in \mathbb{R}$. |
| $t_{dl}^{exec}$ | The deadline of execution of a continuous task $i \in \mathbb{N}$ in one job, where $t_{dl}^{exec} \in \mathbb{R}$. |
| $\tau_{dl}^{exec}$ | The execution period of time for each continuous task $i \in \mathbb{N}$ in one job, starting from $t^0$ and ending at $t_{dl}^{exec}$, where $\tau_{dl}^{exec} > 0$. |
| $\tau_i^{all}$ | An allocation period of time for a continuous task $i$, when this task is running, where $\tau_i^{all} > 0$, $i \in \mathbb{N}$. |
| $\tau_i^{int}$ | An interruption period of time for a continuous task $i$, when this task is not running, where $\tau_i^{int} > 0$, $i \in \mathbb{N}$. |
| $l$ | A counter of the adjoined pairs of interruption-allocation periods, when the interruption period precedes the following allocation period within $\tau_{dl}^{exec}$, where $l \in \mathbb{N}$. |
| $t_{i,l}^{str}$ | The start time of an allocation period $\tau_{i,l}^{all}$ for task $i$, where $t_{i,l}^{str} \in \mathbb{R}$, $i, l \in \mathbb{N}$. |
| $t_{i,l}^{end}$ | The end time of an allocation period $\tau_{i,l}^{all}$ for task $i$, where $t_{i,l}^{end} \in \mathbb{R}$, $i, l \in \mathbb{N}$. |

Continued on the next page

Continued from the previous page

| Symbol | Notation |
|---|---|
| $\tau_{int[i]}^{max}$ | The duration of an interruption for task $i$ which if it is exceeded, leads to the decrease in the increment of the client utility gained for the obtained allocation period in two times from its possible value in the case a client agrees for the GRA's proposal at once after a task has been interrupted, where $\tau_{int[i]}^{max} > 0$, $i \in \mathbb{N}$. |
| $\epsilon_{int[i]}$ | The value for task $i$ which determines the speed of the decrease in the increment of the client utility for the obtained allocation period, influenced by the duration of a single interruption, where $\epsilon_{int[i]} > 0$, $i \in \mathbb{N}$. |
| $\tau_{i,l}^{tot}$ | The total duration of interruptions for task $i$ prior to the allocation period $\tau_{i,l}^{all}$ which is the sum of all those interruption periods, where $\tau_{i,l}^{tot} > 0$, $i, l \in \mathbb{N}$. |
| $\tau_{tot[i]}^{max}$ | The total duration of interruptions for task $i$ within an execution period $\tau_{dl}^{exec}$ which if it is exceeded, leads to the decrease in every increment of the client utility gained for the allocation periods in more than two times from its possible value in the case a client would not take into account the total duration of interruptions, where $\tau_{tot[i]}^{max} > 0$, $i \in \mathbb{N}$. |
| $\epsilon_{tot[i]}$ | The value for task $i$ which determines the speed of the decrease in the increment of the client utility for the obtained allocation period, influenced by the total duration of interruptions, where $\epsilon_{tot[i]} > 0$, $i \in \mathbb{N}$. |
| $\tau_{i,l}^{max}$ | The maximum duration of execution of task $i$, requested by a client from the GRA, while within an interruption period $\tau_{i,l}^{int}$, and its value is limited by the GRA (e.g. $\tau_{i,l}^{max}$ has to be no longer than 7 days), where $\tau_{i,l}^{max} > 0$, $i, l \in \mathbb{N}$. |
| $\tau_{i,l}^{min}$ | The minimum duration of execution of task $i$ which a client is willing to accept from the GRA, while within an interruption period $\tau_{i,l}^{int}$, and it proportionally depends on $\tau_{i,l}^{max}$, where $\tau_{i,l}^{min} > 0$, $i, l \in \mathbb{N}$. |
| $Spec_{i,l}$ | A specification of task $i$ which is submitted to the GRA by a client at the beginning of an interruption period $\tau_{i,l}^{int}$, where $i, l \in \mathbb{N}$. |

Continued on the next page

Continued from the previous page

| **Symbol** | **Notation** |
|---|---|
| $\hat{\tau}_{i,j}^{all}(t)$ | A proposed duration of allocation period for task $i$ in round $j$ at current time $t$ of task execution, where $i, l \in \mathbb{N}$, $t \in \mathbb{R}$. |
| $Pr_{i,j,t}$ | A proposal generated by either negotiating side in respect of its opponent, which contains a task's identifier $i$ and a proposed allocation period $\hat{\tau}_{i,j}^{all}(t)$, where $i, l \in \mathbb{N}$, $t \in \mathbb{R}$. |
| $G_{i,j,l}^{max}(t)$ | The GRA's reservation value which is generated analytically every negotiation round $j$ at time $t$ in our model in order to simulate a negotiation between a client and the GRA, where $i, j, l \in \mathbb{N}$, $t \in \mathbb{R}$. |
| $Value_{i,l}(t)$ | A function of time that calculates a proportion of negotiation interval $\tau_{i,l}^{max} - \tau_{i,l}^{min}$, which is deducted from $\tau_{i,l}^{max}$ to simulate the change of the GRA's reservation value, which has to be smaller or equal to $\tau_{i,l}^{max}$, where $Value_{i,l}(t) \in \mathbb{R}$, $i, l \in \mathbb{N}$, $t \in \mathbb{R}$. |
| $K_1$, $K_2$ | The numbers from the interval $[0, 1]$, which are used to generate $Value_{i,l}(t)$, where $i, l \in \mathbb{N}$, $t \in \mathbb{R}$. |
| $T_{res}$ | A period of the change in the GRA's reservation value as part of the function $Value_{i,l}(t)$, where $T_{res} > 0$. |
| $Random_{i,j,l}$ | A random number which simulates the stochastic deviations in the GRA's reservation value, where $Random_{i,j,l} > 0$, $i, j, l \in \mathbb{N}$. |
| $K_3$ | A percentage of $\tau_{i,l}^{max}$, which determines a standard deviation of the GRA's reservation value, where $K_3 > 0$, $i, l \in \mathbb{N}$. |
| $E(t)$ | An effectiveness function which shows the success in the interval $[0, 1]$ of task execution over time $t$ during an execution period $\tau_{dl}^{exec}$, by increasing linearly during the allocation periods where an effectiveness at the beginning of each period and the angle of its increase are negatively affected by the durations of a single and the total interruptions prior to a corresponding allocation period, where $t \in \mathbb{R}$. |
| $I\left(\tau_{i,l}^{int}\right)$ | A damping function which shows how much the duration of a particular single interruption $\tau_{i,l}^{int}$ affects the effectiveness of task execution for the obtained allocation period $\tau_{i,l}^{all}$, measured by a value from the interval $[0, 1]$, where $i, l \in \mathbb{N}$. |

Continued on the next page

Continued from the previous page

| Symbol | Notation |
|---|---|
| $T\left(\tau_{i,l}^{tot}\right)$ | A damping function which shows how much the total duration of interruptions $\tau_{i,l}^{tot}$ prior to the allocation period $\tau_{i,l}^{all}$ affects the effectiveness of task execution for this allocation period, measured by a value from the interval $[0,1]$, where $i,l \in \mathbb{N}$. |
| $S_{max}$ | The square under the effectiveness function $E(t)$ in case when a task is allocated a period of time which is equal to its execution period $\tau_{dl}^{exec}$ at time $t^0$, which is the best case-scenario for a client, where $S_{max} > 0$, $t \in \mathbb{R}$. |
| $H_{i,l}^{'}$, $H_{i,l}$ | The values of the effectiveness function at the beginning $E\left(t_{i,l}^{str}\right)$ and at the end $E\left(t_{i,l}^{end}\right)$ of the allocation period $\tau_{i,l}^{all}$, where $i,l \in \mathbb{N}$. |
| $S_{sum}^{i}$ | The sum of trapeziums under the effectiveness function for task $i$, which are calculated for the allocation periods, where $S_{sum}^{i} \geq 0$, $i \in \mathbb{N}$. |
| $U_{Cl}^{i}$ | The final utility for task $i$, which is estimated as a sum of all trapeziums $S_{sum}^{i}$ corresponding to the allocation periods in proportion to $S_{max}$, where $U_{Cl}^{i} \in [0,1]$, $i \in \mathbb{N}$. |
| $U_{Cl}^{aver}$ | The average client utility over $N$ tasks in one job, where $N \in \mathbb{N}$. |

## 4.3 ConTask Negotiation Strategy

Our aim is to develop a negotiation strategy for a client to decrease the length of planned and unplanned interruptions while not losing significantly in utility, when executing a task continuously for a long period of time on a Grid. We also assume that the resource availability changes periodically in the Grid and, therefore, the reservation value of the GRA changes periodically as well, because it depends on the resource availability. If the resource availability increases or decreases, then the GRA's reservation value also increases or decreases respectively. Our negotiation strategy for continuous long-term tasks *ConTask* is based on the idea that the client may ask the GRA for a shorter allocation time if this means that the next planned interruption will occur during a period of time when resources are more available. We assume that the GRA always welcomes conceding proposals from a client and that if a client starts negotiation when the GRA's resources are more idle, it will also increase the level of utilisation of these

resources, which is beneficial for the GRA. We also assume that the periodicity of resource availability in the Grid is mostly determined by short term tasks (e.g. on average 7.2 hours in AuverGrid [16]), while the long term tasks are not in majority. However, a Grid's policy is to share all resources among clients and, therefore, no resources can be monopolised for potentially infinite time.

The issue is that a client does not have knowledge about either the change in resource availability, nor the change of the GRA's absolute reservation value over time. Nevertheless, a client can estimate the change of the GRA's reservation value based on the GRA's proposals as it is discussed in Chapter 3. However, this change is estimated in respect of the previous round of negotiation, which shows a tendency in resource availability fluctuation in a short period of time, but it is too vague to learn its periodicity over long periods of time. Hence, we have to choose another estimation to be able to calculate the period of the GRA's reservation value changes to identify the maximum and minimum resource availability in the Grid. The product of negotiation is an allocation period of time obtained for a particular task, and the length of this period of time depends on the resource availability in a Grid. Moreover, a goal of the client is to obtain a longer allocation time and, therefore, the periodicity of allocated time periods over time explicitly displays time intervals which are favourable for a client to negotiate.

Considering the discussion above, an *average allocation period* is chosen as an estimation of the periodicity in resource availability in a Grid (the periodicity of the GRA's reservation value) over time. The average allocation periods increase in the maximum of the GRA's reservation value and decrease in the minimum of the GRA's reservation value, because the reservation value determines how many resources a client can possibly receive. These periods of time are proved to allow to calculate a period of the GRA's reservation value changes and are discussed in Section 4.3.1 together with the algorithm of shortening the allocation period. The ConTask negotiation strategy also includes an evaluation function that allows a client to make a decision online about its choice of tactic, considering the durations of the single and total interruption(s) (accumulating effect) prior to a particular allocation period (see Section 4.3.2).

### 4.3.1 Shortening Task Allocation Periods

We assume that the GRA's reservation value approximately follows a periodicity in resource availability fluctuations, then the closer the GRA's reservation value is to the

maximum of resource availability, the longer the allocation periods that can be assigned to a client's task. Hence, a period of time when the GRA's reservation value is around this maximum is *favourable* for a client to negotiate with the GRA. Assuming this situation, it can be beneficial for a client to ask the GRA for a slightly shorter allocation period of time than the one which would be traditionally accepted in negotiation, if this means that at the beginning of the next interruption period the client is able to negotiate under more favourable conditions (i.e. resources are more available). As a result, a client might obtain a longer allocation period in the next interruption period and this might compensate the loss in utility from shortening its previous allocation period. Moreover, a client can reach an agreement much faster when resources are more available because the GRA is more generous then, i.e. a client may not need to repeat negotiations.

Generally, the client aims to obtain as long an allocation period as possible but it may sacrifice some of its utility in order to gain a higher utility in future, while the GRA would not resist to decrease the allocation period because this follows its interests. In a traditional negotiation, a negotiator usually accepts the proposal of its opponent if this proposal is better than its own in terms of the negotiator's utility. In our work, a client concedes beyond the best of its own or its opponent's proposal in order to gain benefits in future negotiations (e.g. the longer allocation period).

An alternating proposals negotiation protocol, which has been described in Section 3.2.2.1, is modified in this chapter, because the GRA allows a client to change its last best proposal which has been already accepted by the GRA by asking to confirm it. In this modified protocol, a new confirmation message, $CONFIRM$, is introduced which denotes either the GRA's acceptance of the last client's proposal and its request to confirm that proposal or the client's confirmation of its last proposal which has been accepted by the GRA, as depicted in Figure 4.4. That is, a client might confirm its last proposal ($CONFIRM$) or offer a shortened allocation period ($PROPOSAL$) which ends at the maximum of resource availability. The shortened allocation period is considered to be automatically accepted by the GRA as it is shorter than the best client's or its own proposal. Then, the GRA will ask a client to confirm this last proposal (with a shortened allocation period) again. A client can also just accept the current GRA's proposal and send a message $ACCEPT$, which automatically ends a negotiation process. Otherwise, a client can send a new proposal with a shortened allocation period, compared to the last GRA's proposal which would be accepted by a client in a traditional negotiation. This proposal is automatically accepted by the GRA

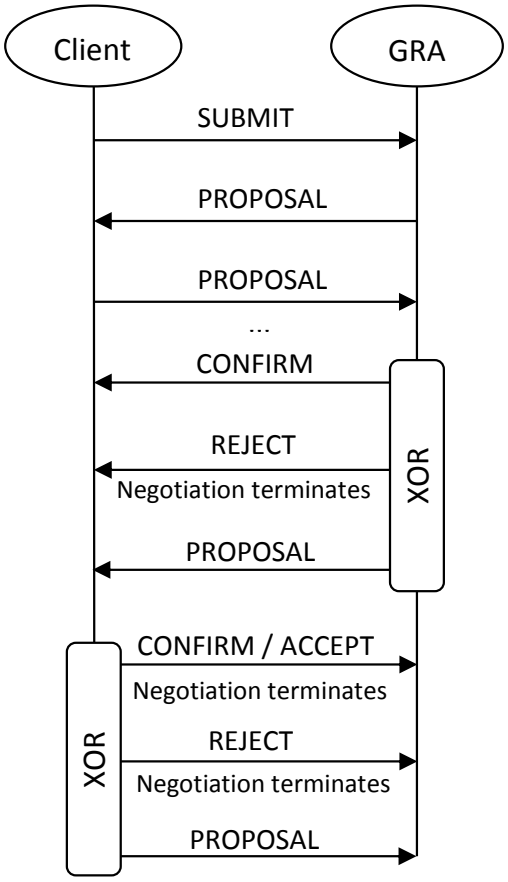



FIGURE 4.4: Modified alternating proposals protocol

as well because it is better than its own possible proposal. Both of the negotiators can also reject ($REJECT$) each other proposals and this message means that a negotiator quits negotiation without reaching a mutual agreement. It has to be noted that all proposals prior to the last (best) one, which might seal an agreement, are generated without considering future negotiation conditions in the following interruptions as they are mostly aimed to obtain as long as possible an allocation period. The following sections describe how a client distinguishes and estimates a favourable period of time to negotiate with the GRA (see Section 4.3.1.1) and how it shortens an allocation period to avoid unfavourable periods of time (see Section 4.3.1.2).

#### 4.3.1.1 Estimation of Favourable Periods of Time

The GRA's reservation value $G^{max}_{i,j,l}(t)$ (see Equation (4.6)), a proportion to the client's maximum value $\tau^{max}_{i,l}$ (see Definition 4.7) for task $i$, changes over negotiation rounds $j$, where each round denotes a virtual second in our simulation. A virtual time means the time scale which we consider in our model, but not the real time. For example, an allocation period is usually measured in hours, while the negotiation process is usually measured in seconds in our work. This measurement allows us to simulate comparably realistic allocation and interruption periods of time by using this scale.[3] In our work, resource availability in a Grid is assumed to change periodically with random deviations (see Section 4.2.2). Here, $G^{max}_{i,j,l}(t)$, compared to Chapter 3, depends also on the current time $t$ as it exhibits a periodicity over time, according to the resource availability fluctuation. If the amount of available resources becomes larger, then the GRA will be more generous in respect of a client, and this will likely to be a beneficial negotiation for a client. If the amount of available resources becomes smaller, then the GRA will be more greedy in respect of a client because of a high probability of resource exhaustion during negotiation, and this will likely to be unsuccessful negotiation for a client. If negotiation fails, then a client has to initiate a negotiation again until its task is assigned an acceptable allocation period. The more generous the GRA and the larger its reservation value, the longer allocation period of time a client can obtain during negotiation. As the GRA's reservation value changes periodically, then the length of the allocation periods will change periodically as well.

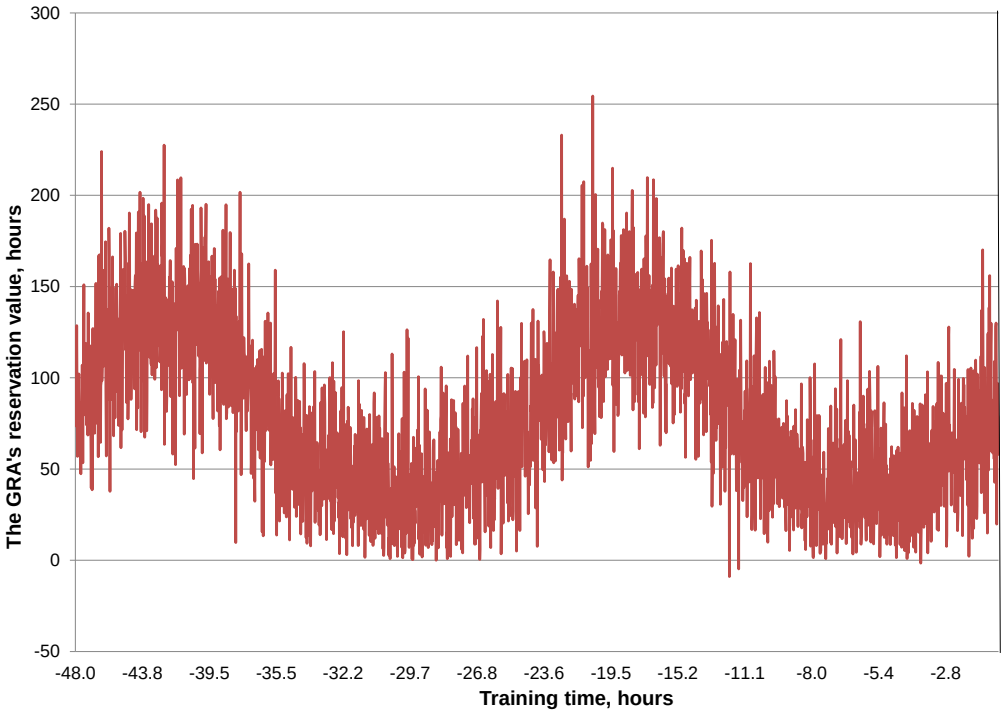
In order to distinguish whether a particular period of time is favourable for negotiation, the client needs to know the average allocation period which can be obtained during this negotiation. Consequently, a client needs some statistics to gather in respect of the average allocation periods over time before launching an execution of its tasks. Therefore, we assume that a client trains first by negotiating with the GRA without launching its tasks even in the case of successful negotiations to calculate the averages of the allocation periods over the training time slots. The training time has to be long enough that a client will be able to estimate the period in which the GRA's reservation value changes, and it is chosen to be the same for all client tasks. Then, a client partitions the total training time on the time slots with a counter $k$ and calculates the average allocation periods over the agreed allocation periods for each time slot.

[3] We use seconds, hours, and other time measurements further in this chapter, but they are all considered to be "virtual" in our simulation model.

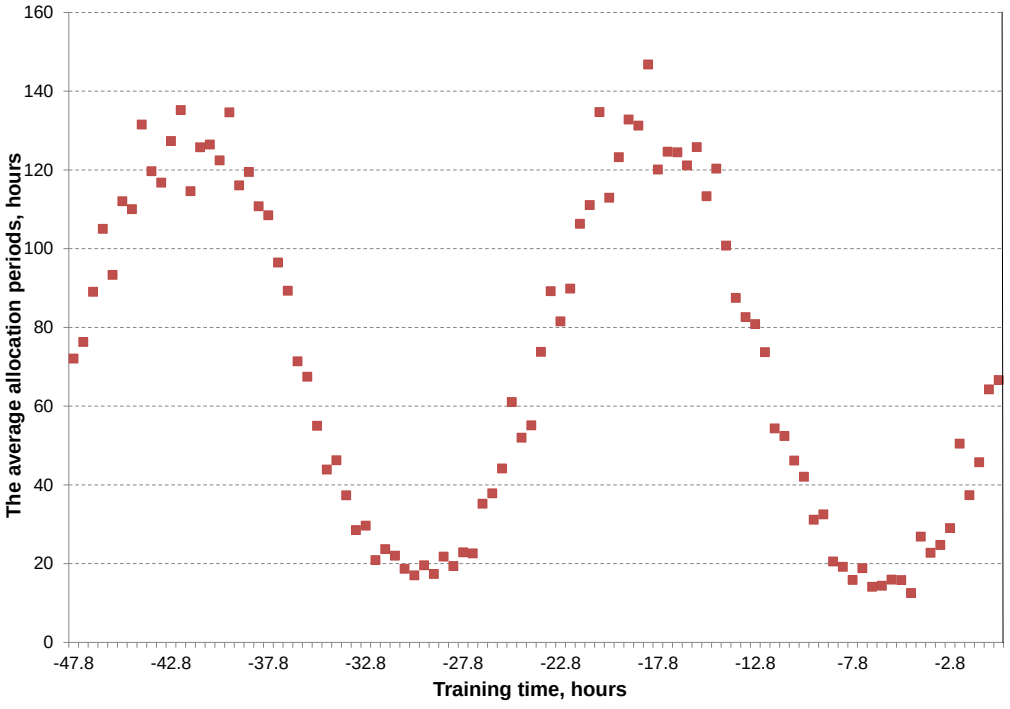
In this way, a client can store a distribution of the obtained *average allocation periods* $\bar{\tau}_i^{all}(t_k)$ for each task $i$ associated with the moments of time $t_k$, where $t_k$ is a mid-point of each training time slot $k$. For example, according to this distribution, a client may know that the average allocation period which can be allocated to its task when the client negotiates with the GRA during a time slot from 5pm and 6pm is 20 hours. The mid-point $t_k$ of this time slot is 5.30pm. We assume that during the training negotiation, a client employs the negotiation strategy with a fuzzy mechanism which takes into account the change in resource availability over negotiation rounds (as described in Chapter 3), but does not shorten its allocation periods. This happens because the aim of training is not to successfully execute client tasks, but to find out which time slots are more or less favourable for a client, and those time slots are expected to be approximately repeated in the future due to the periodicity in resource availability. Therefore, it is necessary for a client to negotiate during unfavourable time slots as well, to be able to recognise them in future.

In our work, we model a training period of time, starting from a smaller negative number which depicts the duration of training, towards zero. When $t^0 = 0$ time unit is reached, then the actual task execution period $\tau_{dl}^{exec}$ starts. The moment of time $t^0$ is chosen to be equal to 0 and this denotes the start of execution for every task. The periodicity of the GRA's reservation value $G_{i,j,l}^{max}(t)$ and of the average allocation periods $\bar{\tau}_i^{all}(t_k)$ obtained by a client during training is simulated and presented in Figures 4.5(a) and 4.5(b) respectively. Figure 4.5(a) shows the change in the GRA's reservation value based on Formula (4.6) with the coefficients $K_1 = 0.5$, $K_2 = 0.65$, $K_3 = 20.0$ and a period $T_{res} = 24.0$ hours. The training period of time is considered to be 48 hours. Figure 4.5(b) demonstrates that the periodic change of the average allocation periods approximately corresponds to a periodicity in the GRA's reservation value. Here, the training time of 48 hours is divided on the 96 time slots of 30 minutes each of them. That is, the set of the allocation periods (an average allocation period) is associated with one time slot $k \in [0, 95]$, $k \in \mathbb{N}$, during which these allocation periods were obtained through negotiation.

According to the distribution of the average allocation periods $\bar{\tau}_i^{all}(t_k)$ over time, a client can choose the most suitable time (the favourable time slot) to negotiate with the GRA after the current allocation period of time will pass. If there is no unexpected interruption due to a resource failure, then we assume that a client cannot negotiate with the GRA until the current allocation period has passed. To find the most suitable time to start negotiation, a client has to approximate the average allocation periods




(a) The change in the GRA's reservation value




(b) The change in the average allocation periods during training

FIGURE 4.5: A demonstration of periodicity in the allocation periods

$\bar{\tau}_i^{all}(t_k)$ obtained during the training time to calculate the period of the GRA's reservation value changes. As long as the length of the allocation periods changes periodically based on the periodic changes in resource availability, we approximate them with a sine-type function $\bar{\tau}_i^{app}(t)$ for task $i$ which depends on time $t$ and it can be presented as in the following equation:

$$\bar{\tau}_i^{app}(t) = A_i \times \sin(B_i \times t) + D_i, \tag{4.14}$$

where $A_i$ is the unknown amplitude of a periodic function, $B_i$ is the unknown angular frequency of this function, $D_i$ is the unknown shift of this function along the $OY$ axis, and $t$ denotes continuous time compared to $t_k$, which indicates discrete time moments within the training period of time. The period of $\bar{\tau}_i^{app}(t)$ is calculated as $T_{app}^i = 2\pi/B_i$ for task $i$, and this period should be approximately equal to $T_{res}$ in case an approximation is accurate enough (see Section 4.2.2). Here, our assumption is that $\bar{\tau}_i^{app}(t) = D_i$ when $t = 0, T_{app}^i/2, T_{app}^i, ...$ A client has to estimate $A_i$, $B_i$ and $D_i$ parameters of a sine-type function based on the training data $\bar{\tau}_i^{all}(t_k)$, where $k$ is the number of the time slot within the training period of time associated with the average allocation period and each training time slot denotes one observation. Consequently, the total number of observations $N_{obs}$ is equal to the total number of time slots within the training period of time (e.g. the 96 observations for an example depicted in Figure 4.5). We indicate $\bar{\tau}_i^{all}(t_k)$ as $\bar{\tau}_{i,k}^{all}$ and $\bar{\tau}_i^{app}(t_k)$ as $\bar{\tau}_{i,k}^{app}$ for simplicity further in the text. A client applies *the least square method* to estimate $A_i$, $B_i$ and $D_i$, and as a result $B_i$ is numerically calculated from the following equation:

$$\begin{aligned}
&\left[\sum_{k=1}^{N_{obs}} \bar{\tau}_{i,k}^{all} \sin(B_i t_k) - 1/N_{obs} \sum_{k=1}^{N_{obs}} \bar{\tau}_{i,k}^{all} \sum_{k=1}^{N_{obs}} \sin(B_i t_k)\right] \times \\
&\left[\sum_{k=1}^{N_{obs}} t_k \sin(B_i t_k) \cos(B_i t_k) - 1/N_{obs} \sum_{k=1}^{N_{obs}} \sin(B_i t_k) \sum_{k=1}^{N_{obs}} t_k \cos(B_i t_k)\right] \\
&+ \left[1/N_{obs} \sum_{k=1}^{N_{obs}} \bar{\tau}_{i,k}^{all} \sum_{k=1}^{N_{obs}} t_k \cos(B_i t_k) - \sum_{k=1}^{N_{obs}} \bar{\tau}_{i,k}^{all} t_k \cos(B_i t_k)\right] \times \\
&\left[\sum_{k=1}^{N_{obs}} \sin^2(B_i t_k) - 1/N_{obs} \left(\sum_{k=1}^{N_{obs}} \sin(B_i t_k)\right)^2\right] = 0,
\end{aligned} \tag{4.15}$$

while $A_i$ is calculated based on the estimated $B_i$ as in the following equation:

$$A_i = \frac{\sum_{k=1}^{N_{obs}} \left(\bar{\tau}_{i,k}^{all} \times \sin(B_i \times t_k)\right) - 1/N_{obs} \sum_{k=1}^{N_{obs}} \left(\bar{\tau}_{i,k}^{all}\right) \times \sum_{k=1}^{N_{obs}} (\sin(B_i \times t_k))}{\sum_{k=1}^{N_{obs}} (\sin^2(B_i \times t_k)) - 1/N_{obs} \left(\sum_{k=1}^{N_{obs}} (\sin(B_i \times t_k))\right)^2}, \tag{4.16}$$

and $D_i$ is calculated based on the known $B_i$ and $A_i$ as in the following equation:

$$D_i = \frac{1}{N_{obs}} \left( \sum_{k=1}^{N_{obs}} \bar{\tau}_{i,k}^{all} - A_i \sum_{k=1}^{N_{obs}} \sin\left(B_i \times t_k\right) \right). \tag{4.17}$$

To explain how the equations for $A_i$, $B_i$ and $D_i$ have been derived, we introduce the following proof.

*Proof.* We apply the least square method to approximate the average allocation periods $\bar{\tau}_{i,k}^{all}$ with a sine-type function $\bar{\tau}_{i,k}^{app}$, which amplitude is $A_i$, angular frequency is $B_i$ and shift along the OY axis is $D_i$. According to this method, we have to minimise a sum of the squared differences between the observed data $\left(t_k, \bar{\tau}_{i,k}^{all}\right)$ and the approximated values $\left(t_k, \bar{\tau}_{i,k}^{app}\right)$ for every observation $k = 0, ..., N_{obs}$, where the counter of observations refers to the counter of time slots within training time, i.e. $k$. The sum of the squared differences $S\left(A_i, B_i, D_i\right)$ is presented in Equation (4.18).

$$S\left(A_i, B_i, D_i\right) = \sum_{k=1}^{N_{obs}} \left( A_i \times \sin\left(B_i \times t_k\right) + D_i - \bar{\tau}_{i,k}^{all} \right)^2. \tag{4.18}$$

Then, according to the least square method, we calculate the partial derivatives of $S\left(A_i, B_i, D_i\right)$ in respect of $A_i$, $B_i$ and $D_i$ and set these equations to zero to find a minimum of this sum.

$$\begin{cases} \sum\limits_{k=1}^{N_{obs}} \left( A_i \times \sin\left(B_i \times t_k\right) + D_i - \bar{\tau}_{i,k}^{all} \right) \times \sin\left(B_i \times t_k\right) = 0, \\ \sum\limits_{k=1}^{N_{obs}} \left( A_i \times \sin\left(B_i \times t_k\right) + D_i - \bar{\tau}_{i,k}^{all} \right) \times A_i \times \cos\left(B_i \times t_k\right) \times t_k = 0, \\ \sum\limits_{k=1}^{N_{obs}} \left( A_i \times \sin\left(B_i \times t_k\right) + D_i - \bar{\tau}_{i,k}^{all} \right) = 0, \end{cases} \tag{4.19}$$

Finally, Equations (4.15)-(4.17) can be derived from the system of equations presented in Formula (4.19). □

After we have estimated $A_i$, $B_i$ and $D_i$, we can present the approximated function $\bar{\tau}_i^{app}\left(t\right)$ (see Equation (4.14)) of the average allocation periods over time. The peaks of this function denote the periods of time when a client can obtain the longer allocation

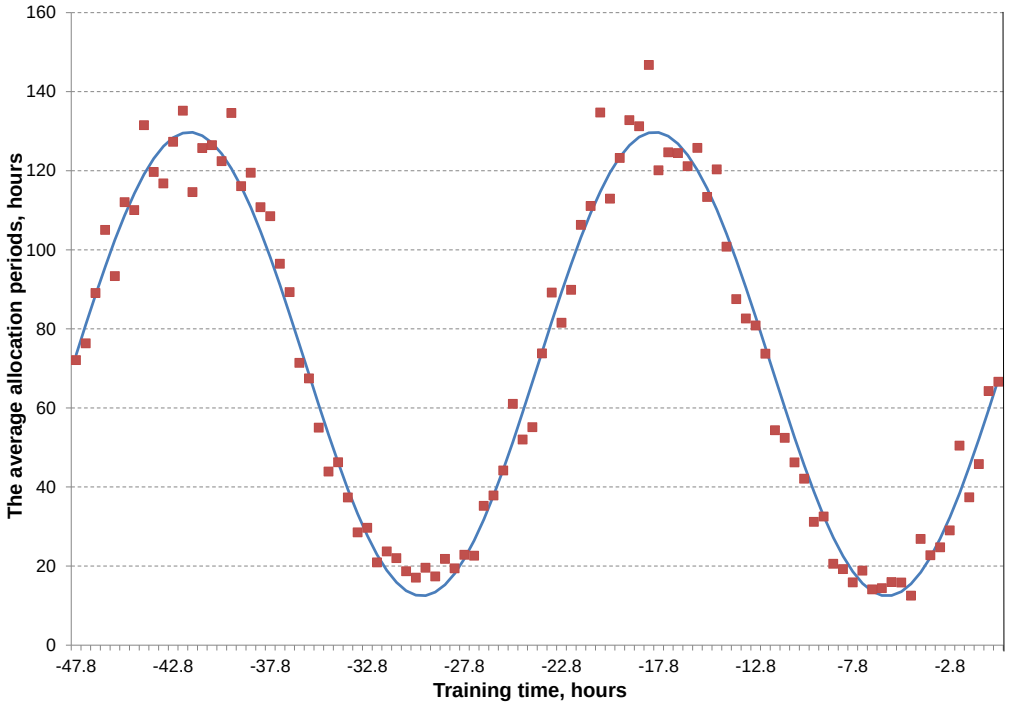



FIGURE 4.6: The approximated sine-type function $\bar{\tau}_i^{app}(t)$

periods and an agreement can be reached faster (see Figure 4.6). The lows of this function denote the periods of time when a client can obtain the shorter allocation periods and an agreement may not be reached with the larger probability because of resource exhaustion (see Figure 4.5(a) where the GRA's reservation values become negative). In other words, any period of time which is closer to the maximum of $\bar{\tau}_i^{app}(t)$ is favourable for a client, while this one which is closer to the minimum of $\bar{\tau}_i^{app}(t)$ is unfavourable. In this way, the maximum of $\bar{\tau}_i^{app}(t)$ denotes the maximum of the GRA's reservation value and, consequently, the maximum of resource availability in a Grid. Figure 4.6 demonstrates an approximated sine-type function for the average allocation periods presented with a continuous curve. Here, the approximated sine-type function is $\bar{\tau}_i^{app}(t) = 58.7575 \times \sin(0.262296 \times t) + 71.3278$ with a period $T_{app}^i = 23.94$ hours (the period of the GRA's reservation value change in Figure 4.5(a)).

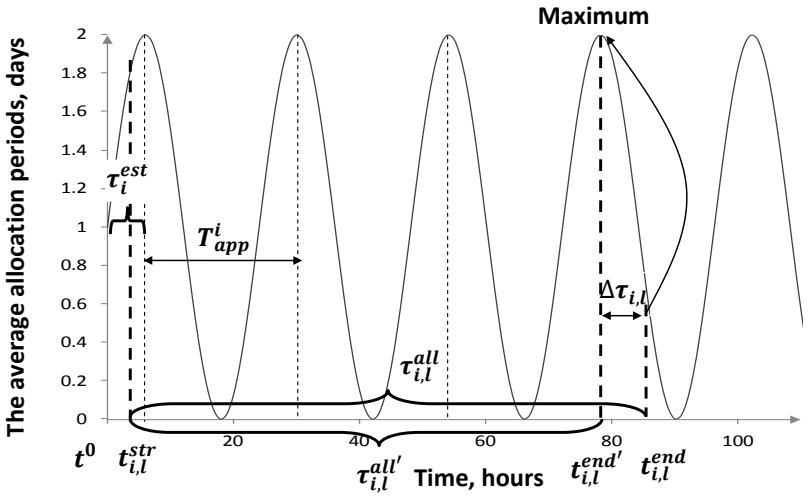



FIGURE 4.7: A mechanism to shorten an allocation period $\tau_{i,l}^{all}$

#### 4.3.1.2 Shortening of an Allocation Period

A client has to decide whether to shorten an allocation period before accepting the GRA's proposal with the aim to negotiate within a favourable time period when the next interruption occurs. A client makes this decision at the end of negotiation, when its next proposal will be worse than the current GRA's proposal, or when the GRA has accepted the client's last proposal and asked to confirm it. If the client decides to shorten an allocation period, then another issue regards how much this allocation period should be shortened. A client generally intends to receive as long allocation period of time as possible. Therefore, if a client decides to shorten an allocation period, it has to be as a small shortening as possible which, nevertheless, allows a client to negotiate within a favourable period of time when task terminates.

Assume that the time period on which a client shortens its allocation period is $\Delta\tau_{i,l}$, and the shortened allocation period is $\tau_{i,l}^{all'} = \tau_{i,l}^{all} - \Delta\tau_{i,l}$. The end time of the shortened allocation period $\tau_{i,l}^{all'}$ is $t_{i,l}^{end'}$, while the end time of the non-shortened allocation period $\tau_{i,l}^{all}$ is $t_{i,l}^{end}$. The shortened allocation period $\tau_{i,l}^{all'}$ has to end around the maximum of resource availability, and this maximum has to be the closest one to the end of the non-shortened allocation period $\tau_{i,l}^{all}$. In this way, a client tries to avoid unnecessary loss in utility by shortening an allocation period $\tau_{i,l}^{all}$ only on the duration of time $\Delta\tau_{i,l}$, which is required to shift $t_{i,l}^{end}$ to the closest maximum of resource availability $t_{i,l}^{end'}$, i.e. $\Delta\tau_{i,l} << \tau_{i,l}^{all}$ (see Figure 4.7).

Figure 4.7 shows how $\Delta\tau_{i,l}$ is estimated, considering the period $T^i_{app}$ of a sine-type function $\bar{\tau}^{app}_i(t)$ where this function is used by a client to identify the peaks of resource availability, as depicted in this figure. To estimate $\Delta\tau_{i,l}$, the client deducts a duration of time $\tau^{est}_i$ between the start of task execution $t^0$ and the first approximate maximum of resource availability from $t^0$ (derived algorithmically from $\bar{\tau}^{all}_i(t_k)$) and the specific number of periods $n_{i,l}$ to reach the closest maximum of $\bar{\tau}^{app}_i(t)$ in respect of $t^{end}_{i,l}$. If $t^{end}_{i,l}$ is at the maximum of $\bar{\tau}^{app}_i(t)$, then $\Delta\tau_{i,l}$ is equal to zero. Assuming a client is aware of $\bar{\tau}^{app}_i(t) = D_i$ when $t = 0, T^i_{app}/2, T^i_{app}, ...$, then a client is able to estimate $\Delta\tau_{i,l}$ as presented in the following equation:

$$\Delta\tau_{i,l} = \left(t^{end}_{i,l} - t^0\right) - \left(\tau^{est}_i + T^i_{app} \times n_{i,l}\right), \tag{4.20}$$

where $t^0 = 0$ denotes the beginning of an execution period $\tau^{exec}_{dl}$ as described in Section 4.2.1 (it does not mean the moment of time when a task has been launched, but the moment of time when the first negotiation has been initiated by a client for this task) and in our model it is equal to zero (i.e. a client starts the initial negotiation for all tasks at the same time). In Figure 4.7, the closest maximum in respect of $t^{end}_{i,l}$ is indicated with the word "Maximum". As $\Delta\tau_{i,l}$ has to be always a positive value, then we can estimate the number $n_{i,l}$ of periods $T^i_{app}$, which have to be deducted, as presented in the following equations:

$$\left(t^{end}_{i,l} - t^0\right) - \left(\tau^{est}_i + T^i_{app} \times n_{i,l}\right) > 0, \tag{4.21}$$

and

$$n_{i,l} < \frac{t^{end}_{i,l} - t^0 - \tau^{est}_i}{T^i_{app}}, \tag{4.22}$$

then

$$n_{i,l} = int\left(\frac{t^{end}_{i,l} - t^0 - \tau^{est}_i}{T^i_{app}}\right), \tag{4.23}$$

where $int(\cdot)$ denotes a function which rounds $n_{i,l}$ towards the smaller integer part. It has to be noted that if the allocation period is short and there is no maximum of $\bar{\tau}^{app}_i(t)$ within this period, a client does not shorten it because in this case $\Delta\tau_{i,l}$ is longer than $\tau^{all}_{i,l}$, i.e. $\tau^{all'}_{i,l} < 0$ ($\tau^{all'}_{i,l}$ always should be a positive value). Consequently, when $\tau^{all'}_{i,l} < 0$, then a client accepts $\tau^{all}_{i,l}$ instead. Finally, a decision making algorithm for a client, considering the shortening of an allocation period, is presented below.

Algorithm 4.2 is an extension of the client decision making algorithm for negotiation,

**Algorithm 4.2** Client algorithm to shorten $\tau_{i,l}^{all}$

1: {A client is going to accept the GRA's proposal or confirm its own proposal to the GRA, and this proposal is $\tau_{i,l}^{all}$}
2: Calculate $T_{app}^{i} = 2\pi/B_i$ (see Section 4.3.1.1)
3: Calculate $n_{i,l}$ (see Formula (4.21))
4: Calculate $\Delta\tau_{i,l}$ (see Formula (4.20))
5: **if** $\Delta\tau_{i,l} < \tau_{i,l}^{all}$ **then**
6: Calculate $\tau_{i,l}^{all'} = \tau_{i,l}^{all} - \Delta\tau_{i,l}$
7: Calculate $t_{i,l}^{end'} = t_{i,l}^{end} - \Delta\tau_{i,l}$ {The end time is re-calculated to start the next interruption period at the correct point in time}
8: Send a new proposal of $\tau_{i,l}^{all'}$ to the GRA
9: Respond to the request of confirmation to the GRA {Agreement for $\tau_{i,l}^{all'}$}
10: **else**
11: Accept allocation period $\tau_{i,l}^{all}$ {Agreement for $\tau_{i,l}^{all}$}
12: **end if**

described in Chapter 3. Specifically, this is an expansion of line 32 in Algorithm 3.1 regarding when and how the agreement is reached. Previously, it has been assumed that a client always accepts the GRA's proposal (e.g. $\tau_{i,l}^{all}$), when the client cannot offer a better proposal or when the client's negotiation deadline $t_{dl}^{c}$ is reached. We have also assumed that the client's $t_{dl}^{c}$ and the GRA's $t_{dl}^{gra}$ negotiation deadlines are equal and when one of the deadlines is reached, a client accepts the GRA's last proposal. That is, the line 32 of Algorithm 3.1 shows only the case when a negotiation finishes successfully, while the line 5 shows the case when negotiation fails due to resource exhaustion. Hence, the ending of negotiation in this algorithm is trivial as opposed to Algorithm 4.2, where a negotiation process ends not just when one of the negotiators accepts its opponent's last best proposal, but also when the client decides to shorten this last best proposed allocation period before confirming its decision to the GRA.

**Example for Algorithm 4.2.**

This example shows the end of negotiation between a client and the GRA, where a client may shorten its allocation period if necessary. Here, $\tau_i^{est} = 21600.0$ (virtual seconds). The duration of allocation period is $\tau_{i,l}^{all} = 196710.0$ (virtual seconds), and this allocation period has been accepted by either side, but not confirmed by a client yet. The value of $\Delta\tau_{i,l}$ is calculated as in Formula (4.20).

$T_{app}^{i} = 86400.0$ (virtual seconds);

$n_{i,l} = 2$;

$\Delta\tau_{i,l} = 196765.0 - (21600.0 + 2 * 86400.0) = 2365.0$ (virtual seconds);

As long as $2365.0 < 196710.0$, then the client decides to shorten the allocation period 196710.0 (virtual seconds), and the shortened allocation period is 194345.0 (virtual seconds), which the client confirms to the GRA.

The other example is when the client decides not to shorten the allocation period, because this allocation period is too short and it does not contain at least one maximum of resource availability. In this example, the duration of allocation period is $\tau_{i,l}^{all} =$ 66362.3 (virtual seconds).

$T_{app}^{i} = 86400.0$ (virtual seconds);

$n_{i,l} = 23$;

$\Delta\tau_{i,l} = 2075200.0 - (21600.0 + 23 * 86400.0) = 66400.0$ (virtual seconds);

As long as $66400.0 > 66362.3$, then the client decides not to shorten the allocation period 66362.3 (virtual seconds), and confirms this duration to the GRA.

### 4.3.2 Addressing Task Interruption Periods

As we discussed in Section 4.2, a duration of interruption for a continuous task negatively affects the client utility by decreasing it when this duration becomes longer. If one negotiation fails, than a client has to start another negotiation process. Each single negotiation is nominally limited to a specific number of rounds, but generally a negotiation process continues until an agreement is reached with the GRA. Ideally, if a client negotiates when resource availability is at its maximum, the duration of a single interruption is usually no longer than the maximum duration of a single negotiation (i.e. a maximum number of negotiation rounds). In other words, a client does not need to repeat negotiations because of resource exhaustion and, therefore, the risk of long interruptions which may significantly affect the client utility is not large. However, an approximation of the average allocation periods of time, described in Section 4.3.1, provides a client with a period $T_{app}^{i}$, which approximately corresponds to the real period $T_{res}$ of the GRA's reservation value changes. Moreover, unexpected interruptions e.g., a resource failure, may also prevent a client from negotiating during favourable time.

Consequently, a client may encounter cases when it has to negotiate under a lack of resources. In these cases, there is a high risk of significantly long interruptions, because negotiation might be repeated a large number of times. We assume that the impact of the duration of interruption on the client utility changes gradually. At the beginning of an interruption, the impact increases with a smaller speed, while the longer this interruption, the faster this speed, and this change at some point can be sharp and fast. Therefore, a client has to reason whether it is close in its negotiation to the point in time when its utility starts to be affected significantly by the duration of interruption. Starting from this point in time, a client has to become more generous and to reach an agreement faster with the GRA in order to avoid a significant decrease in utility. If the impact of the duration of interruption is still insignificant, then the client can be less generous to get a longer allocation period of time. The significance of the impact has to be judged by the client, according to the nature of its task.

In Chapter 3, we have developed an evaluation function $Eval_{i,j}(j_x)$ (see Equation (3.37)) which has allowed a client to decide which tactic (i.e. the level of greediness) to choose in negotiation round $j$ for task $i$ to take into account the risk of resource exhaustion during negotiation, based on the average speed and overall direction of the changes in resource availability estimated over previous negotiation rounds. If there is a high risk of resource exhaustion before the end of single negotiation, than a client chooses that level of greediness $\beta^c_{i,j}$ (see Section 3.3.3.3), which leads to the closest expected round of agreement $j_x$ among all expected agreement rounds, produced by the different combinations of the fuzzy membership functions. If there is no such risk, then the client's most desirable level of greediness will correspond to farthest expected agreement round $j_x$. In the following sections, we introduce two additional components of this evaluation function, which reflect an impact of the current single and total durations of interruption on the effectiveness of task execution (see Section 4.3.2.1). We also demonstrate the flexibility of our evaluation function in terms of embedding new components (see Section 4.3.2.2).

#### 4.3.2.1 Evaluation Function Additional Components

In Section 4.2.3, the damping functions $I(\tau^{int}_{i,l})$ (see Equation (4.7)) and $T(\tau^{tot}_{i,l})$ (see Equation (4.8)) are described, which estimate the impact of the duration of a single interruption and the total duration of all interruptions prior to a particular allocation period on the effectiveness of task execution during this allocation period. Here, we

consider similar functions $\breve{I}\left(\breve{\tau}_{i,l}^{int}(t)\right)$ and $\breve{T}\left(\breve{\tau}_{i,l}^{tot}(t)\right)$ which describe the impact of the duration of current single, $\breve{\tau}_{i,l}^{int}(t)$, and total, $\breve{\tau}_{i,l}^{tot}(t)$, interruptions on the following allocation period. These functions are different from $I(\tau_{i,l}^{int})$ and $T(\tau_{i,l}^{tot})$ because they include a time dependence $t$. The functions $\breve{I}\left(\breve{\tau}_{i,l}^{int}(t)\right)$ and $\breve{T}\left(\breve{\tau}_{i,l}^{tot}(t)\right)$ are presented in the following equations:

$$\begin{aligned}\breve{I}\left(\breve{\tau}_{i,l}^{int}(t)\right) &= \frac{1}{e^{\left(\breve{\tau}_{i,l}^{int}(t)-\tau_{int[i]}^{max}\right)/\epsilon_{int[i]}}+1},\\ \breve{T}\left(\breve{\tau}_{i,l}^{tot}(t)\right) &= \frac{1}{e^{\left(\breve{\tau}_{i,l}^{tot}(t)-\tau_{tot[i]}^{max}\right)/\epsilon_{tot[i]}}+1},\end{aligned} \tag{4.24}$$

where $(\tau_{int[i]}^{max}, \epsilon_{int[i]})$ and $(\tau_{tot[i]}^{max}, \epsilon_{tot[i]})$ describe how fast the possible increment in the client's utility for the upcoming allocation period decreases, following the increase of a single and total interruptions (see Definitions 4.4, 4.2, 4.6 and 4.3). The closer the length of interruption to $\tau_{int[i]}^{max}$ and $\tau_{tot[i]}^{max}$, the more significant the impact of the interruption duration on the client's utility, while the values of $\epsilon_{int[i]}$ and $\epsilon_{tot[i]}$ determine a speed of decrease in the client's utility towards zero. Consequently, if the duration of interruption is too close to $\tau_{int[i]}^{max}$ and / or $\tau_{tot[i]}^{max}$, a client may become more generous in negotiation with the GRA in order to avoid exceeding those durations, which should be reflected in its evaluation function (see Section 3.3.3.3). Generally, our idea is that a client should be able to decrease the duration of interruptions by conceding more towards the GRA and reaching an agreement faster in this way, if these durations threaten to affect significantly its utility. Here, we also attempt to balance between the client's aim to obtain a longer allocation period and the risk of prolonged interruptions.

Assume that a client defines the *sensitivity thresholds* $\chi_i^{int}$ and $\chi_i^{tot}$ with respect to a single and total interruptions for task $i$ respectively. As soon as the difference between the value of any damping function, i.e. $\breve{I}\left(\breve{\tau}_{i,l}^{int}(t)\right)$ or $\breve{T}\left(\breve{\tau}_{i,l}^{tot}(t)\right)$, and its largest possible value 1 at time $t$ becomes larger than $\chi_i^{int}$ for $\breve{I}\left(\breve{\tau}_{i,l}^{int}(t)\right)$ or $\chi_i^{tot}$ for $\breve{T}\left(\breve{\tau}_{i,l}^{tot}(t)\right)$, the client concludes that the interruption is long enough to significantly affect its utility. In Figure 4.8, the damping function $\breve{I}(\breve{\tau}_{i,l}^{int}(t))$, which is simulated with the parameters $\tau_{int[i]}^{max} = 110.0$ seconds and $\epsilon_{int[i]} = 10.0$ seconds, starts visibly falling towards zero when $\chi_i^{int} = 0.01$. So, if we assume that the client's sensitivity threshold $\chi_i^{int}$ for a single interruption is 0.01, then we have to deduct approximately 4.5 values of $\epsilon_{int[i]}$ from $\tau_{int[i]}^{max}$ to identify the beginning of the time range where the duration of a single interruption starts significantly affecting the client's utility. Formally, $\chi_i^{int}$ or $\chi_i^{tot}$ denotes the difference between the largest possible value of the corresponding damping function

e.g., $\breve{I}(0) \approx 1$, (no impact on client utility) and the values of these functions when the durations of the current single $\tau_{i,l}^{int}(t)$ and / or total $\tau_{i,l}^{tot}(t)$ interruptions exceed the durations $\left(\tau_{int[i]}^{max} - m_i \times \epsilon_{int[i]}\right)$ and $\left(\tau_{tot[i]}^{max} - h_i \times \epsilon_{tot[i]}\right)$ as shown in Equation 4.25.

$$\begin{aligned} \chi_i^{int} &= 1 - \breve{I}\left(\tau_{int[i]}^{max} - m_i \times \epsilon_{int[i]}\right), \\ \chi_i^{tot} &= 1 - \breve{T}\left(\tau_{tot[i]}^{max} - h_i \times \epsilon_{tot[i]}\right), \end{aligned} \tag{4.25}$$

where $m_i, h_i \in \mathbb{R}$ denotes the number of the values of $\epsilon_{int[i]}$ or $\epsilon_{tot[i]}$ which are deducted from $\tau_{int[i]}^{max}$ or $\tau_{tot[i]}^{max}$ to identify the start of the time range where the client utility is significantly affected. Considering $\chi_i^{int}$ and $\chi_i^{tot}$ are determined by a client, we can estimate the values of $m_i$ and $h_i$ from Equations (4.25) and (4.7) as in Equation (4.26).

$$\begin{aligned} m_i &= \ln\left(\left(1 - \chi_i^{int}\right)/\chi_i^{int}\right). \\ h_i &= \ln\left(\left(1 - \chi_i^{tot}\right)/\chi_i^{tot}\right). \end{aligned} \tag{4.26}$$

It has to be noted that in our current work $\chi_i^{int}$ and $\chi_i^{tot}$ are chosen to be the same for all tasks.

In Chapter 3, our evaluation function has only one component $Cm_{i,j}$, which takes into account the risk of resource exhaustion before the end of negotiation for task $i$ in round $j$. The new components $Cm_{i,l}^{int}(t)$ and $Cm_{i,l}^{tot}(t)$, described in this section, solve the distinct problem of prolonged interruption durations. $Cm_{i,l}^{int}(t)$ denotes whether the length of the current single interruption is in the time range where the utility is significantly affected by its duration, i.e. $\breve{\tau}_{i,l}^{int}(t) > \tau_{int[i]}^{max} - m_i \times \epsilon_{int[i]}$, while $Cm_{i,l}^{tot}(t)$ has the same idea for the current total interruption, and they are presented in the following formulas:

$$\begin{aligned} Cm_{i,l}^{int}(t) &= \left(\tau_{int[i]}^{max} - m_i \times \epsilon_{int[i]}\right) - \breve{\tau}_{i,l}^{int}(t), \\ Cm_{i,l}^{tot}(t) &= \left(\tau_{tot[i]}^{max} - h_i \times \epsilon_{tot[i]}\right) - \breve{\tau}_{i,l}^{tot}(t). \end{aligned} \tag{4.27}$$

The function $Cm_{i,l}^{int}(t)$ or $Cm_{i,l}^{tot}(t)$ produces negative values when $\breve{\tau}_{i,l}^{int}(t) > \tau_{int[i]}^{max} - m_i \times \epsilon_{int[i]}$ or $\breve{\tau}_{i,l}^{tot}(t) > \tau_{tot[i]}^{max} - h_i \times \epsilon_{tot[i]}$ and positive values when $\breve{\tau}_{i,l}^{int}(t) < \tau_{int[i]}^{max} - m_i \times \epsilon_{int[i]}$ or $\breve{\tau}_{i,l}^{tot}(t) < \tau_{tot[i]}^{max} - h_i \times \epsilon_{tot[i]}$ respectively, where negative values indicate that a duration of interruption has exceeded the corresponding sensitivity threshold.

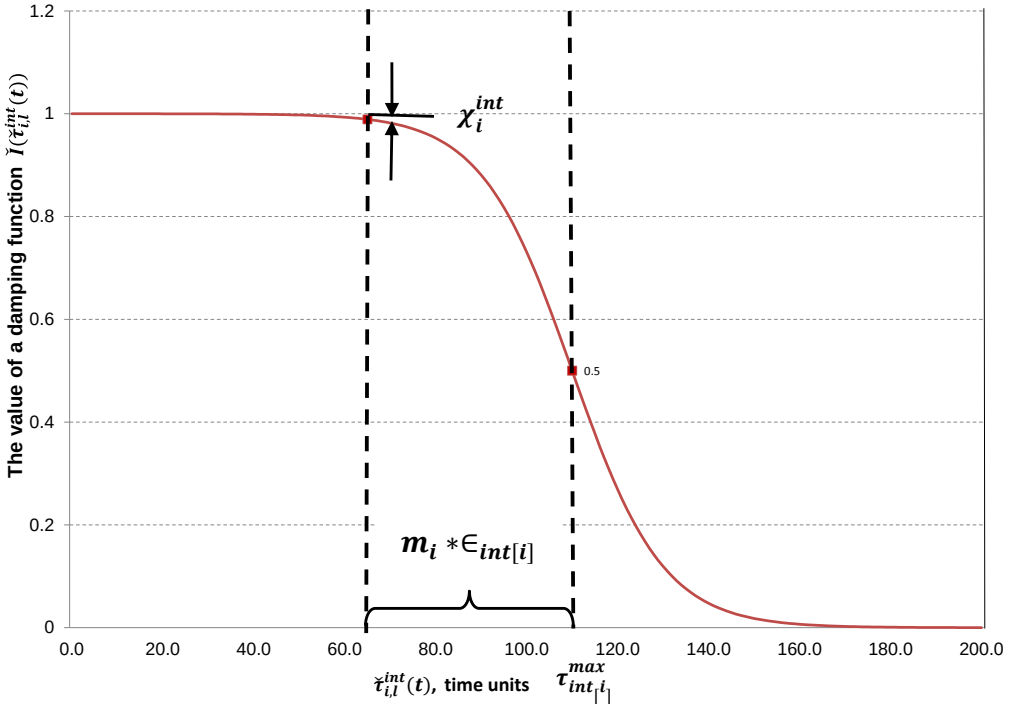



FIGURE 4.8: A simulation of the damping function $\breve{I}(\breve{\tau}_{i,l}^{int}(t))$

#### 4.3.2.2 Extended Evaluation Function

In Chapter 3, the evaluation function $Eval_{i,j}(j_x)$ (see Equation (3.37)) estimates the risk of resource exhaustion for each task $i$ in every negotiation round $j$ for each expected round of agreement $j_x$ if a client chooses a particular level of greediness $\beta_{i,j}^c$. The maximum of this function for the different expected $j_x$ indicates the smallest or the largest level of greediness among all levels, produced by the different fuzzy sets' combinations (see Section 3.3.3.3). Therefore, this function consists of the two summands, where only one summand at time $t$ can have a non-zero value: if $j_x/t_{dl}^c$ is maximised, the longest negotiation is chosen (the largest level of greediness); if $(t_{dl}^c - j_x)/t_{dl}^c$ is maximised, then the shortest negotiation is chosen (the smallest level of greediness) within a nominal negotiation deadline $t_{dl}^c$ (see Section 3.3.3.3) of a single negotiation. A zero or non-zero value of a summand is determined by the Heaviside step function $\theta(\cdot)$, where the component function is its argument. The step function $\theta(\cdot)$ is equal to zero for the negative arguments, and it is equal to one for the positive or zero arguments.

The extended evaluation function $Eval''_{i,j,l}(j_x,t)$, as compared to $Eval_{i,j}(j_x)$, also depends on the current time $t$ as it evaluates the durations of the current single and total interruptions. It also has one more index, $l$, which denotes the number of single interruption within the execution duration $\tau_{dl}^{exec}$. In addition to the component $Cm_{i,j}$, this function includes the components $Cm_{i,l}^{int}(t)$ and $Cm_{i,l}^{tot}(t)$ as the arguments of the two more Heaviside step functions.Then, a client is able to assess not just the risk of resource exhaustion every negotiation round $j$, but also the risk of too long interruptions which can significantly affect its utility. The new Heaviside step functions can be embedded into evaluation function as the factors to the existing step function as depicted in the following equation:

$$Q_{i,j,l}(t) = \theta(Cm_{i,j})\,\theta\left(Cm_{i,l}^{int}(t)\right)\theta\left(Cm_{i,l}^{tot}(t)\right), \tag{4.28}$$

where $Q_{i,j,l}(t)$ denotes a function of those factors, which can have a value of either zero or one at time $t$ in round $j$. Here, we show that our evaluation function is flexible in terms of embedding new components, and its modified version $Eval''_{i,j,l}(j_x,t)$ is described in the following equation:

$$Eval''_{i,j,l}(j_x,t) = \left(\frac{t_{dl}^c - j_x}{t_{dl}^c}\right)(1 - Q_{i,j,l}(t)) + \left(\frac{j_x}{t_{dl}^c}\right)Q_{i,j,l}(t). \tag{4.29}$$

The proof of this modification is discussed below.

*Proof.* As explained in Chapter 3, if the risk of resource exhaustion is high, then $\theta(Cm_{i,j}) = 0$ in $Eval_{i,j}(j_x)$. In our new evaluation function $Eval''_{i,j,l}(j_x,t)$, we also consider that if the length of interruption(s) is in the range where a client loses significantly in its utility, then $\theta\left(Cm_{i,l}^{int}(t)\right) = 0$ and $\theta\left(Cm_{i,l}^{tot}(t)\right) = 0$. In other words, if at least one of the Heaviside step functions is equal to zero in $Q_{i,j,l}(t)$, it means that a client may lose significantly in utility, fail negotiation, etc. Therefore, a client has to become generous when at least one of these step functions is equal to zero. That is, the summand $\left(\frac{t_{dl}^c - j_x}{t_{dl}^c}\right)$ which refers to the closest expected agreement round $j_x$ should be maximised in this case, i.e. the shortest negotiation has to be chosen. If only all Heaviside step functions are equal to one, then the longest negotiation has to be chosen. Here, we can compose a modified intermediate evaluation function $Eval'_{i,j,l}(j_x,t)$ by using only one new component $\theta\left(Cm_{i,l}^{int}(t)\right)$ and the previous version of evaluation

function $Eval_{i,j}(j_x)$ as in the following equation:

$$Eval'_{i,j,l}(j_x,t) = \left(\frac{t^c_{dl}-j_x}{t^c_{dl}}\right) \times \left(1-\theta\left(Cm^{int}_{i,l}(t)\right)\right) + Eval_{i,j}(j_x) \times \theta\left(Cm^{int}_{i,l}(t)\right). \tag{4.30}$$

Then, the shortest negotiation $max_{j_x}\left(\left(t^c_{dl}-j_x\right)/t^c_{dl}\right)$ is estimated every time when $\theta\left(Cm^{int}_{i,l}(t)\right)=0$ and the longest negotiation $max_{j_x}\left(j_x/t^c_{dl}\right)$ can be only considered if $\theta\left(Cm^{int}_{i,l}(t)\right)=1$ and $\theta\left(Cm_{i,j}\right)=1$. In the same way, we can compose the final modified evaluation function $Eval''_{i,j,l}(j_x,t)$ by using one more new component $\theta\left(Cm^{tot}_{i,l}(t)\right)$ and the modified intermediate evaluation function $Eval'_{i,j,l}(j_x,t)$ as presented in Equation (4.31).

$$Eval''_{i,j,l}(j_x,t) = \left(\frac{t^c_{dl}-j_x}{t^c_{dl}}\right) \times \left(1-\theta\left(Cm^{tot}_{i,l}(t)\right)\right) + Eval'_{i,j,l}(j_x,t) \times \theta\left(Cm^{tot}_{i,l}(t)\right). \tag{4.31}$$

If we open the brackets in Equation (4.31), considering $Eval'_{i,j,l}(j_x,t)$ and $Eval_{i,j}(j_x)$, it will be transformed into Equation (4.29). In this way, we have shown the flexibility of our evaluation function $Eval''_{i,j,l}(j_x,t)$ which can be enhanced by embedding new Heaviside step functions with other arguments (objectives) as factors to the existing step functions. □

TABLE 4.2: List of notation for Section 4.3

| Symbol | Notation |
|---|---|
| $k$ | The counter of time slots on which the training time is divided by a client to estimate the average allocation (interruption) periods over each slot, where $k \in \mathbb{N}$. |
| $t_k$ | The mid-point of time slot $k$ on within a training period of time, where $t_k \in \mathbb{R}$. |
| $\bar{\tau}^{all}_i(t_k)$ | A distribution of the average allocation periods, calculated by a client for each time slot $k$ within a training period of time for task $i$, where $i,k \in \mathbb{N}$, $t_k \in \mathbb{R}$. |
| $\bar{\tau}^{app}_i(t)$ | An approximation of a distribution of the average allocation periods over continuous time t, presented as a sine-type function for task $i$, where $i \in \mathbb{N}$, $t \in \mathbb{R}$. |

Continued on the next page

Continued from the previous page

| Symbol | Notation |
|---|---|
| $A_i$ | An amplitude of $\bar{\tau}_i^{app}(t)$ for task $i$, which shows the difference between the peak value of an allocation period and its middle value, where $A_i \in \mathbb{R}$, $i \in \mathbb{N}$. |
| $B_i$ | An angular frequency of $\bar{\tau}_i^{app}(t)$ for task $i$, from which a period of $\bar{\tau}_i^{app}(t)$ can be derived, where $B_i \in \mathbb{R}$, $i \in \mathbb{N}$. |
| $D_i$ | A shift of $\bar{\tau}_i^{app}(t)$ for task $i$ along the $OY$ axis, where $D_i \in \mathbb{R}$, $i \in \mathbb{N}$. |
| $T_{app}^i$ | An approximated period of $\bar{\tau}_i^{app}(t)$ for task $i$, which should correspond to the period of the GRA's reservation value changes $T_{res}$, where $i \in \mathbb{N}$. |
| $N_{obs}$ | The number of observations used for approximation, which refers to the number of time slots inside the training period of time, $N_{obs} \in \mathbb{N}$. |
| $S(A_i, B_i, D_i)$ | The sum of the squared differences between the observed average allocation periods and the allocation periods produced by an approximated sine-type function. This sum has to be minimised to obtain an accurate approximation, according to the least square method, where $S(A_i, B_i, D_i) \geq 0$. |
| $\Delta\tau_{i,l}$ | The duration of time on which an allocation period $\tau_{i,l}^{all}$ can be shortened to start the next negotiation in a favourable period of time, where $i, l \in \mathbb{N}$. |
| $\tau_{i,l}^{all'}$ | The shorterned allocation period of time in respect of $\tau_{i,l}^{all}$, which finishes at the moment of time $t_{i,l}^{end'}$ for task $i$, where $i, l \in \mathbb{N}$. |
| $\tau_i^{est}$ | The duration of time between $t^0$ and the first approximate maximum of resource availability for task $i$, where $i \in \mathbb{N}$. |
| $n_{i,l}$ | The number of periods $T_{app}^i$, which has to be deducted from $t_{i,l}^{end} - t^0$ to calculate $\Delta\tau_{i,l}$, where $n_{i,l}, i, l \in \mathbb{N}$. |
| $\breve{\tau}_{i,l}^{int}(t)$, $\breve{\tau}_{i,l}^{tot}(t)$ | The current durations of a single and total interruption(s) for task $i$ respectively, where $i, l \in \mathbb{N}$. |
| $\chi_i^{int}$, $\chi_i^{tot}$ | The sensitivity thresholds of a client in respect of a single and total interruptions for task $i$ respectively, where $i \in \mathbb{N}$. |

Continued on the next page

Continued from the previous page

| Symbol | Notation |
|---|---|
| $m_i$, $h_i$ | The number of $\epsilon_{int[i]}$ and $\epsilon_{tot[i]}$ which is deducted from $\tau^{max}_{int[i]}$ and $\tau^{max}_{tot[i]}$ respectively to identify the time ranges, where the client's utility is significantly affected by the duration of interruption(s), where $m_i, h_i \in \mathbb{R}$, $i \in \mathbb{N}$. |
| $Cm^{int}_{i,l}(t)$ | A function which shows to a client whether the threshold $\chi^{int}_i$ has been crossed at time $t$ for the current single interruption $\breve{\tau}^{int}_{i,l}(t)$, where $Cm^{int}_{i,l}(t) \in \mathbb{R}$, $t \in \mathbb{R}$, $i, l \in \mathbb{N}$. |
| $Cm^{tot}_{i,l}(t)$ | A function which shows to a client whether the threshold $\chi^{tot}_i$ has been crossed at time $t$ for the current total interruption $\breve{\tau}^{tot}_{i,l}(t)$, where $Cm^{tot}_{i,l}(t) \in \mathbb{R}$, $t \in \mathbb{R}$, $i, l \in \mathbb{N}$. |
| $\theta(\cdot)$ | The Heaviside step function, which is equal to zero, when its argument is negative, and it is equal to one, when its argument is positive or zero. |
| $Q_{i,j,l}(t)$ | A component function which includes all factors, affecting client's decision at time $t$ in negotiation round $j$, where $Q_{i,j,l}(t) = \{0, 1\}$, $t \in \mathbb{R}$, $i, j, l \in \mathbb{N}$. |
| $Eval''_{i,j,l}(j_x, t)$ | The modified evaluation function, compared to $Eval_{i,j}(j_x)$, which considers the durations of a single and total interruption(s). It signals to a client when the client's utility can be significantly affected in the case of the prolonged interruptions, where $Eval''_{i,j,l}(j_x, t) \in [0, 1]$, $t \in \mathbb{R}$, $i, j, l, j_x \in \mathbb{N}$. |

## 4.4 Results and Discussion

The aim of section is to evaluate the client's utility if the client uses the ConTask negotiation strategy, compared to the negotiation strategy discussed in Chapter 3, which does not take into account continuity of task execution. We also test our ConTask strategy for different Grid environment conditions (e.g. the amplitude of resource availability changes), which are reflected in the change of the GRA's reservation value as discussed in Section 4.2.2. The GRA's reservation value $G^{max}_{i,j,l}(t)$ changes periodically, according to Formula (4.6), where $K_3$ determines the standard deviation of this reservation value. The GRA's reservation value may have a larger or smaller deviations, which reflects

its level of predictability for a client and accounts to the dynamism of the clients and resources in a Grid as well as its own policy. For example, large deviations may mean that the large number of the short-term tasks require a large amount of resources at once, but they also release them relatively soon. However, small deviations may denote that the resources are not significantly demanded, or the tasks mostly require long-term executions. As the deviations of the GRA's reservation value may significantly affect an outcome of negotiation, we consider them as an important factor for evaluation. That is, we compare the client utilities for the cases when the standard deviation of the GRA's reservation value can be 1%, 5%, 10% and 20% of the client's maximum value (see Definition 4.7). Here, 1% denotes an extremely small deviation, while 20% differ from 1% by more than one order of magnitude and, therefore, it has been chosen as a significantly larger deviation. The client utility is monotonically decreasing for the chosen deviations, and it is expected to decrease for the larger deviations, because of the increasing non-determinism in resource availability fluctuation. Larger deviations than 20% are less realistic, because they might mean that several resource intensive tasks occupy a major part of Grid resources.

We also assume that each single negotiation takes at most 100 rounds, which is equivalent to 100 virtual seconds and this parameter has been adopted from the previous chapter. In our model, all time units are modelled in the virtual time units, not a real time. The maximum values $\tau_{i,l}^{max}$ of the allocation period for the different tasks are generated randomly from the range of $7 \pm (1-2)$ virtual days, i.e. the duration of time in advance of which the GRA can predict its resource availability with acceptable accuracy in order to commit to any resource allocation [15], while the execution period $\tau_{dl}^{exec}$ for all tasks is equal to one virtual year. In this way, a task has to expect at least 48 planned interruptions, where a client may have to repeat negotiations until it reaches an agreement with the GRA. The period of the change $T_{res}$ in the GRA's reservation value is considered to be 24 virtual hours. The client utilities are averaged over 100 tasks, which are assumed to be executed independently. The attributes for each task $i$ are considered to be $\tau_{int[i]}^{max} = 100.0$, $\tau_{tot[i]}^{max} = 4800.0$, $\epsilon_{int[i]} = 30.0$ and $\epsilon_{tot[i]} = 2000.0$ virtual seconds. These attributes are chosen experimentally, and they generally aim not to significantly affect utility, if a client negotiates successfully during 100 rounds without negotiation repetitions. More repetitions and, as a result, longer interruption periods, affect client utility more significantly. Consequently, if the total duration of interruption exceeds the planned total duration $48 \times 100$ of interruption, this will also significantly affect the client's utility. The values $\epsilon_{int[i]}$ and $\epsilon_{tot[i]}$ are chosen to ensure

the client's sensitivity towards the durations of interruption, i.e. the functions $I(\cdot)$ and $T(\cdot)$ start visibly falling towards zero from the beginning of the corresponding interruptions (see Equations (4.7), (4.8)).

The following Section 4.4.1 evaluates the client's utility when applying an algorithm of shortening a would-be agreed allocation period during the current negotiation in order to negotiate at the maximum of resource availability in the next interruption period. Then, Section 4.4.2 evaluates the client's utility when applying a modified evaluation function, which takes into account the durations of a single and total interruption(s), while negotiating for a particular allocation period. Finally, Section 4.4.3 discusses the client's utility when applying the whole ConTask negotiation strategy for the cases of planned and unexpected task interruptions.

### 4.4.1 Shortening Task Allocation Periods

In this section, we compare the cases "wShort" when a client uses the shortening algorithm (see Algorithm 4.2) in order to correct a would-be outcome of negotiation with the cases "noShort" when this algorithm is not used (see Chapter 3). The client utilities are also compared for the idealistic "noTrain" and realistic "wTrain" scenarios, where a client can estimate precisely (i.e. ideal) or approximately (i.e. real) the period of change $T_{res}$ in the GRA's reservation value as depicted in Figure 4.9. An approximate estimation of this period is implemented as discussed in Section 4.3.1.1, and it depends on the random deviations in the GRA's reservation value from its periodic dependency where the larger deviations denote the less deterministic GRA's behaviour. All simulations are also conducted in respect of the different amplitudes of the changes in the GRA's reservation value, determined by a coefficient $K_2$ in Equation (4.4). A larger amplitude potentially allows a client to obtain longer allocation periods around the maximum of resource availability, but it also means a larger risk of resource exhaustion around the minimum of resource availability. In this evaluation, we have chosen $K_2 = 0.85$ as a large amplitude of $G^{max}_{i,j,l}(t)$, while $K_2 = 0.65$ as a small amplitude of $G^{max}_{i,j,l}(t)$. The value of $K_1$ is equal to 0.5 in all cases.

Figure 4.9 shows the change in the client utilities, depending on the deviations of the GRA's reservation value for the different scenarios. First, it has to be noted that all cases "wShort" show better utilities than the cases "noShort", where the utilities calculated under the idealistic scenario are higher than all other utilities. Second, it is

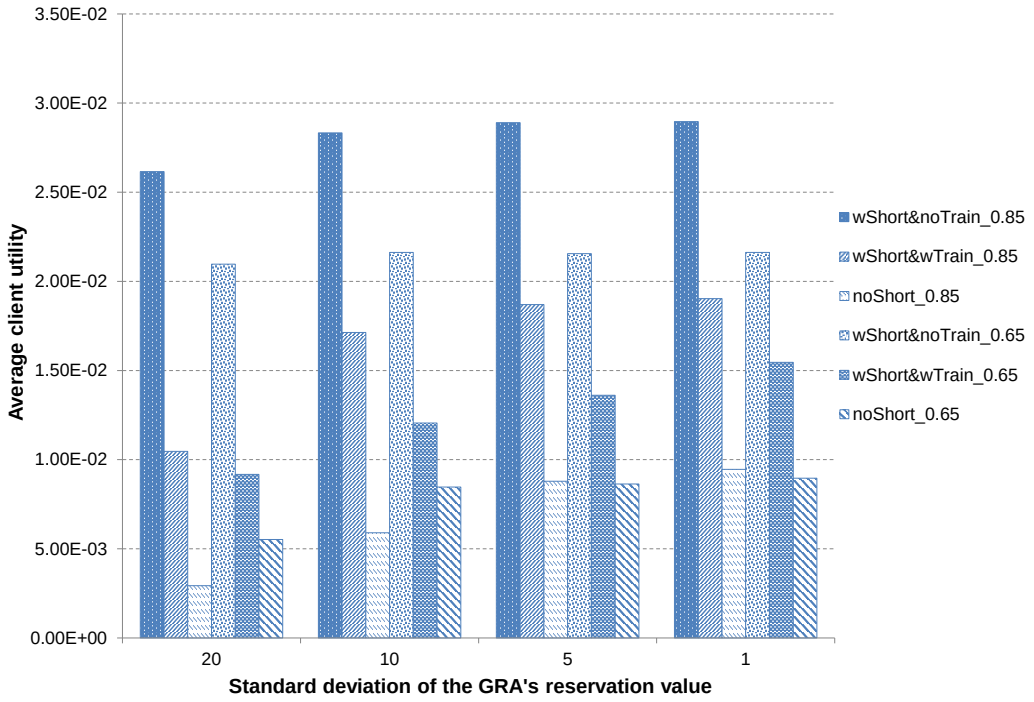



FIGURE 4.9: The client utility with the shortening algorithm

important to mention that the larger amplitude of the GRA's reservation value leads to the larger utilities for the cases "wShort" (e.g. "wShort&wTrain_0.85" compared to 'wShort&wTrain_0.65"), while generally the lower utilities for the cases "noShort" (i.e. "noShort_0.85" compared to "noShort_0.65"). It is also shown that the distinct amplitudes have less affect on the cases "noShort" for lower deviations of the GRA's reservation value, because the risk of resource exhaustion is larger for higher deviations and the "noShort" strategy does not attempt to negotiate around the maximum of resource availability which might happen only by accident. That is, a client which employs the "noShort" negotiation strategy might often negotiate far from the maximum of resource availability, in which case the larger amplitude leads to the larger risk of resource exhaustion. Hence, the negotiation strategy "wShort" shows the larger utilities for the amplitude 0.85 than 0.65, because a client mostly negotiates around the maximum of resource availability where the larger amplitude means the larger GRA's reservation values and, as a result, the longer allocation periods. The larger deviations of the GRA's reservation value can also hide the difference between the amplitudes as

for the cases "wShort&wTrain_0.85" and 'wShort&wTrain_0.65". It has to be noted that this evaluation shows the larger differences in the utilities between the idealistic and realistic cases for the larger deviations, which means that the larger deviations lead to the less accurate estimation of $T_{res}$ by a client. The deviations almost not affect the idealistic cases as they do not depend on the estimation of this period.

### 4.4.2 Addressing Task Interruption Periods

In this section, we evaluate how an extended evaluation function "wEval" (see Equation (4.29)), which additionally takes into account the durations of a single and total interruption, improves the client's utility, compared to the case when our initial evaluation function (see Equation (3.37)) is used, "noEval". Here, in all cases the amplitude $K_2$ of the GRA's reservation value fluctuation is equal to 0.85 as we aim to check whether our ConTask strategy is effective in respect of reducing the durations of interruption, which is more essential for the larger amplitudes. The duration of interruptions can be reduced in terms of the more conceding client, which can reach an agreement faster with the GRA. However, the larger concessions also result in the shorter allocation periods. Therefore, a client has to balance between its aim to reach an agreement faster and to obtain the longer allocation period, which is reflected in its level of sensitivity in respect of the duration of interruption. The smaller sensitivity threshold, i.e. $\chi_i^{int}$ and $\chi_i^{tot}$ (see Formula (4.25)), means a more generous client, while the larger threshold means a less generous client.

Hence, we compare the client utilities for the cases of the different sensitivity thresholds, which values are the same for a single and total interruptions for generalisation. The sensitivity threshold is chosen to be equal to 0.01 for the case "wShort&wEval_0.01", 0.1 for the case "wShort&wEval_0.1", etc., which are depicted in Figure 4.10. Figure 4.10 demonstrates the client utilities, obtained by using our evaluation function (extended or initial) for the different deviations of the GRA's reservation value. Here, the smallest threshold is 0.01 and the largest is 0.5, where 0.01 denotes that a client always tends to be generous and 0.5 denotes that a client mostly tends to be greedy. It also has to be noted that the shortening algorithm, evaluated in Section 4.4.1, is not used in any of these cases "noShort".

This figure shows that the smaller sensitivity thresholds almost always lead to the better utilities for a client. In this way, this evaluation supports an idea that in the

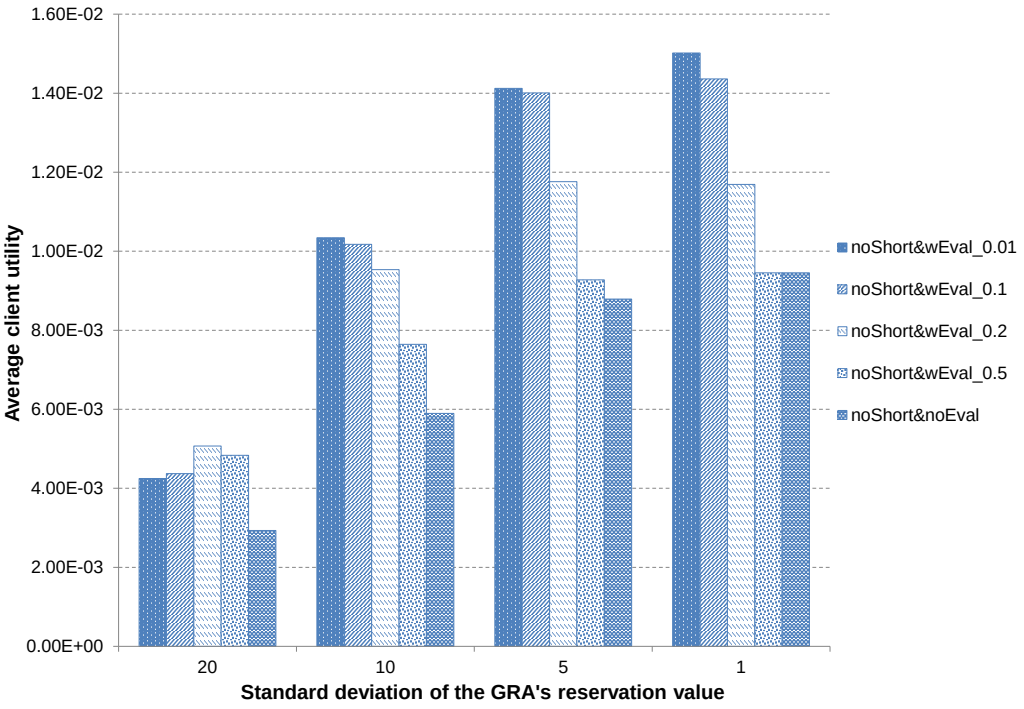



FIGURE 4.10: The client utility with an extended evaluation function to consider interruptions

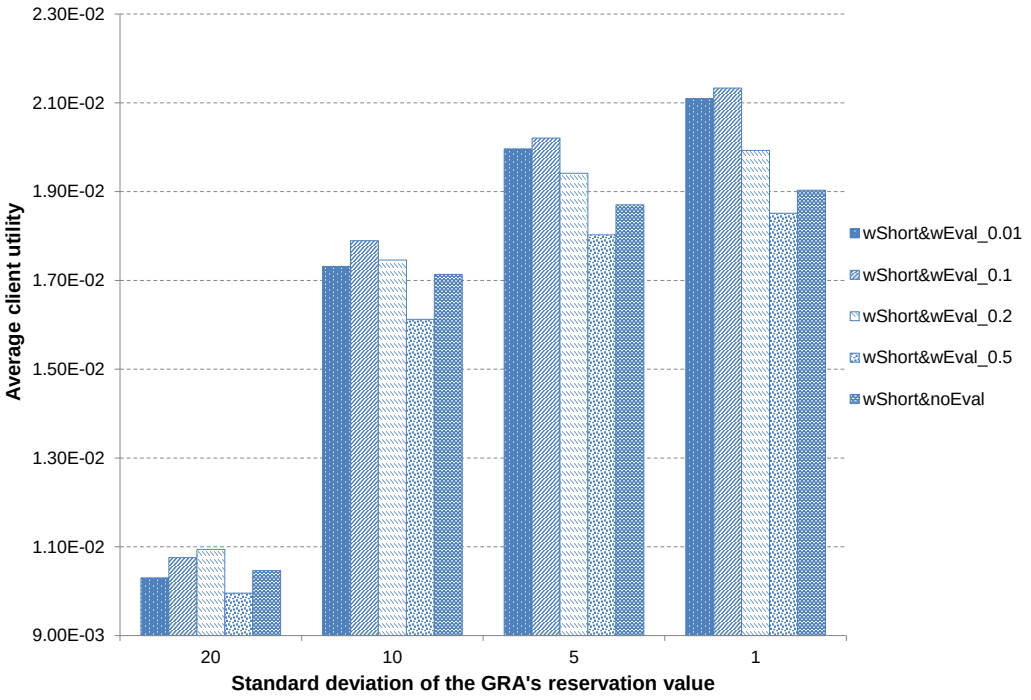
cases when a client cannot negotiate under favourable conditions (a client does not use the shortening algorithm), it is beneficial to be more generous in respect of the GRA. However, the difference between the utilities obtained for the different thresholds diminishes towards the larger deviations of the GRA's reservation value. The case with the large deviation, i.e. 20%, shows the best utility for the sensitivity threshold 0.2. This can be explained by a controversial impact of the large deviations on the client's utility, where they significantly increase a possibility to obtain longer allocation periods due to the large positive deviations and at the same time increase a risk of resource exhaustion due to the large negative deviations. Hence, the less predictable Grid environments require more balanced sensitivity thresholds in terms of the variation between generosity or greediness, and all straight-forward strategies, i.e. predominantly generous or greedy, are less beneficial for a client in these cases.

### 4.4.3 Applying the ConTask Negotiation Strategy

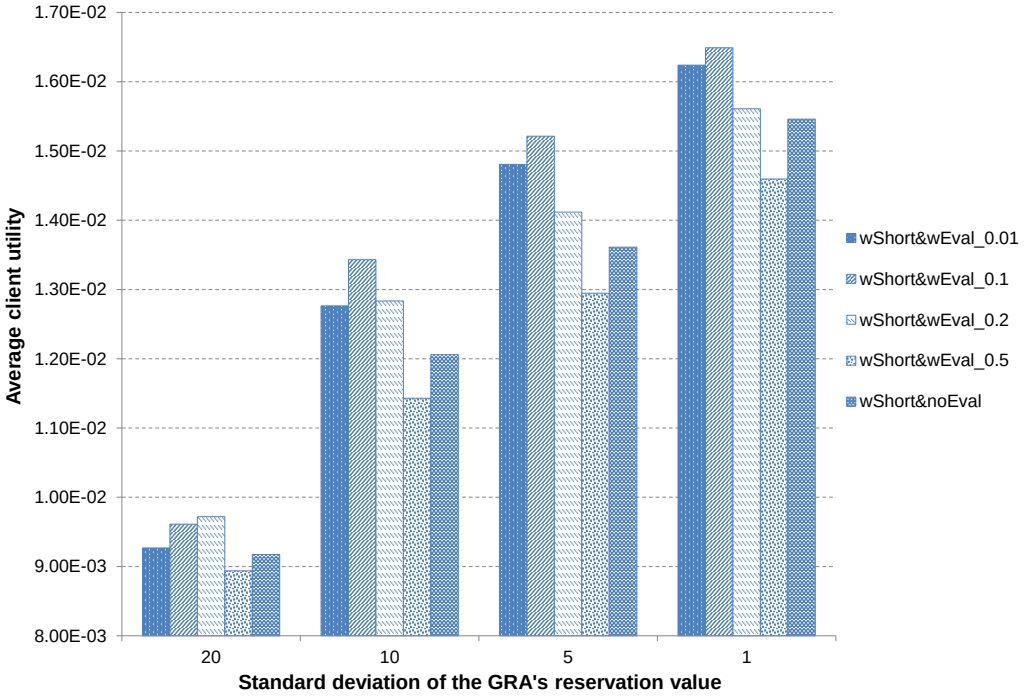
In this section, we discuss the client utilities gained when the ConTask negotiation strategy is used by a client, "wShort&wEval", i.e. the shortening algorithm and an extended evaluation function, compared to the cases when it is not used, "noShort&noEval". We also compare the strategies which use the different sensitivity thresholds and their impact on the client's utility in combination with the shortening algorithm. Here, we also test our strategy in respect of the different probabilities of the unexpected task interruptions.

Figure 4.11 shows the client utilities for the different deviations of the GRA's reservation value with the different amplitudes (i.e. 0.85 and 0.65) for the cases when the ConTask strategy is used by a client. This figure does not show the cases "noShort&noEval" as they produce significantly smaller utilities than any of the depicted corresponding cases, which are presented in Figure 4.9 as "noShort_0.85" for a high amplitude and "noShort_0.65" for a small amplitude. It also has to be mentioned that the high amplitude of the GRA's reservation value (see Figure 4.11(a)) leads to the smaller differences between the utilities for the different sensitivity thresholds than in the case of the small amplitude (see Figure 4.11(b)), which is the result of the larger variance in the GRA's reservation value. Note that almost all utilities, produced with the ConTask strategy in Figure 4.11, are larger than the utilities produced only with the shortening algorithm "wShort&noEval" or the extended evaluation function (see Figure 4.10). In other words, we can conclude that the shortening algorithm in combination with the extended evaluation function benefits a client more than just each of those contributions separately, except for the cases with sensitivity threshold 0.5 and some cases with sensitivity threshold 0.01. It means that the extreme cases of generosity and greediness diminish a positive impact of the shortening algorithm on the client's utility as the large greediness extends negotiation for no necessity and the large generosity shortens negotiation too much when a longer allocation period can be obtained.

Figure 4.12 demonstrates the client's utilities for the cases when in addition to the planned interruptions at the end of each allocation period, the unexpected interruptions are also considered with the probability 0.2 for the cases "wL" and 0.5 for the cases "wH". Here, the ConTask negotiation strategy "wShort&wEval" is compared to the cases when this strategy is not used "noShort&noEval" or only an extended evaluation function is used "noShort&wEval". In this evaluation, a sensitivity threshold for the cases "wEval" is chosen to be 0.1 as it has showed the best result for almost all cases in




(a) The client utility for a high amplitude of $G_{i,j,l}^{max}(t)$




(b) The client utility for a small amplitude of $G_{i,j,l}^{max}(t)$

FIGURE 4.11: The client utility when the ConTask strategy is used by a client

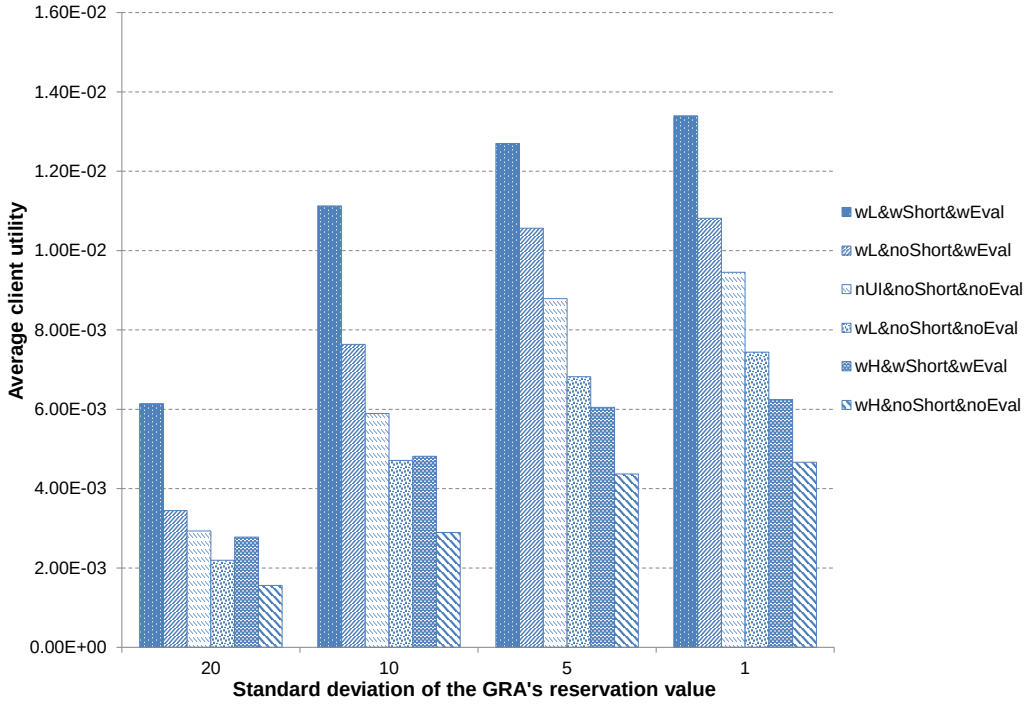



FIGURE 4.12: The client utility when the unexpected task interruptions may occur

Figure 4.11. In Figure 4.12, our ConTask negotiation strategy "wShort&wEval" shows the better utilities than our previous negotiation strategy "noShort&noEval" (Chapter 3), which has not been specifically developed for continuous tasks. For the cases of the low probability of task unexpected interruption, our ConTask strategy outperforms even the case with no unexpected interruption "nUI&noShort&noEval".

## 4.5 Conclusions

In this chapter, we have described the novel ConTask negotiation strategy for long-term continuous task execution, which is based on the negotiation strategy described in Chapter 3. ConTask strategy includes a shortening algorithm, which allows a client to shorten an allocation period proposed or accepted by the GRA before confirming it. This algorithm is applied at the end of negotiation when a client is going to accept the GRA's proposal or the GRA has accepted a client's proposal. A client may decide to

slightly shorten a would-be traditionally agreed allocation period in order to start next interruption period when resources are more available. The new strategy also improves the evaluation function we introduced in Chapter 3, and it is used by the client every round of negotiation to choose the level of greediness, considering not only a risk of resource exhaustion but also the durations of interruptions.

Our simulation results show that our ConTask negotiation strategy improves the client's utility in almost all simulated cases, compared to our previous negotiation strategy (see Chapter 3), which has not been specifically designed for continuous long-term tasks. The shortening algorithm leads to better utilities when the period of the GRA's reservation value fluctuation can be estimated with the larger accuracy (in particular, in the cases of the smaller deviation of this value). Our extended evaluation function benefits a client in almost all cases, where the smaller sensitivity thresholds are generally more beneficial for a client, but a bit larger threshold (not extremely large) can be more beneficial for a client in the cases of the more random resource availability fluctuations (in particular, in the cases of the large deviations of the GRA's reservation value). Generally, our ConTask negotiation strategy shows a larger improvement in the client's utility when the shortening algorithm is used with an extended evaluation function. This strategy outperforms our previous strategy in almost all cases with or without unexpected task interruptions.

# Chapter 5

# Dynamic Task Re-allocation for Simultaneous Tasks

## 5.1 Introduction

Recently much research focuses on smart systems which, for example, control climate parameters inside a building, or monitor the level of pollution in the environment [30]. Previously, we have assumed that tasks are independent in terms of input data, i.e. they do not need to wait for the results from other tasks to be able to run without interruptions. As an example, assume that a flat in a building is monitored in respect of the climate changes (e.g. local temperatures, directions of air and heat currents), and there are three tasks (e.g. a system of non-stationary bulk partial differential equations of fluid mechanics) which analyse these changes in the different rooms of this flat. To estimate the climate changes in the flat, we have to aggregate the results from all these three tasks, which are processed by another task. In this way, a task which is responsible for the estimation of the climate changes in the flat has to wait for the results from the other three tasks which calculate the climate changes in the rooms over time. Consequently, the task which depends on the other tasks in terms of data cannot be executed independently. This example shows one of the possible dependencies between tasks.

The tasks which acquire the results from other tasks, *recipient-tasks*, receive a sequence of data in real time. Each recipient-task is expected to receive the required input

data from a particular *sender-task* every specific time unit (e.g. every second), while processing previous data. If this data is not received in time, then the client's system will produce erroneous results to some degree, i.e. the longer this delay, the larger the probability that the last received data from a sender-task is significantly different from the current data that would be received. A client, who knows the nature of its tasks, can approximately predict the time during which an estimated parameter e.g., temperature, may change by an amount significant enough for the actuators e.g., heating actuators, to be activated. As time has passed since a sender-task stopped sending its data to a particular recipient-task, the recipient-task has to stop as well in order to not produce wrong commands for the actuators. This time is called a *delay time*, and during this time the recipient-task can be run with the last received data (before interruption of the corresponding sender-task(s)) from the interrupted sender-tasks until this time has passed. The length of the delay time depends on the number of stopped / interrupted sender-tasks and the level of impact of their data on the results produced by a recipient-task. A larger number of stopped sender-tasks means less control of the parameter estimated by the recipient-task, and the delay time is shorter for this recipient-task. A smaller impact of the sender-task data on the recipient-task results means this sender-task does not significantly affect the parameter, and the delay time is longer for this recipient-task.

*Stopped* and *interrupted* states are two different states for a task, where the first means that a task is not running but has the resources it requires to execute, while the second state denotes that a task is not running because it lost its resources. When a task is stopped or interrupted, it does not receive or send any data. This means that all sender-tasks that send data to this interrupted task directly or indirectly are also stopped (they have no one to send its data), and all recipient-tasks just use the last received data from this interrupted task until the end of their delay times. Our work aims to consider and resolve situations when tasks do not produce their results in time, but have interruptions which affect the execution of other tasks. We propose to re-allocate resources from one client's task, which has resources, to another one, which is interrupted, if the duration of interruption is too long. The tasks can be re-allocated only within one job (see Definition 3.1), submitted by a client in order to avoid an unauthorised re-allocation request. The proposed *SimTask re-allocation strategy* allows a client to decide when to re-allocate resources for a particular interrupted task and which task has to donate these resources in the cost of its own utility. A final decision about re-allocation is negotiated with the GRA, because a client does not

have a direct access to resources. The GRA is considered to be generally content to re-allocate resources among client's tasks, because a client does not ask for new resources. However, the GRA might offer a shorter period of resource utilisation to the interrupted task than the one allocated to the donor-task, because a task re-allocation is considered as an additional service (i.e. it has not been agreed in the initial agreement) and this can be regarded as a "payment" within our model.

To conclude the discussion above, we define *simultaneous* tasks as tasks for which it is desirable to be executed over the same period of time, because of their dependencies in terms of the data exchange. These dependencies can be *implicit*, when the tasks are not connected directly, but their work affects the same task. For example, in the case of monitoring climate parameters inside the building, the tasks which estimate an average temperature near the floor and the ceiling over time are not directly dependent, but their work affects the resulting average temperature in the room estimated by other task. The dependencies between tasks can also be *explicit* when one task is directly connected to another one by means of sending or receiving the data. For example, in the same case of monitoring climate parameters, the task which estimates an average temperature near the floor of the room has to be run simultaneously with the task which estimates the average temperature in the whole room. This chapter consists of Section 5.2 which formalises the dependencies among tasks, Section 5.3 which discusses the SimTask re-allocation strategy applied by a client to resolve task interruptions, Section 5.4 which describes the evaluation of this SimTask strategy, and Section 5.5 which concludes this chapter. Sections 5.2 and 5.3 include tables of notations, which are introduced in these sections (see Tables 5.1 and 5.2).

## 5.2 Formal Model

In Chapters 3 and 4, we have considered a situation where a client submits multiple tasks to a Grid, but they were all independent in terms of their execution. In this chapter, we consider the situations where tasks are inter-dependent in terms of data exchange, and we formalise these dependencies. The dependencies among tasks also mean that the interruption of one task will affect the utilities of dependent tasks as well as its own utility, which is considered in our modified client utility function. In this section, we formalise a task as well as its dependencies with other tasks (see Section

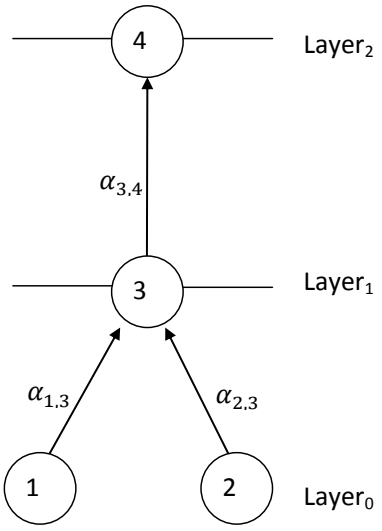



FIGURE 5.1: A dependence between the tasks $i = 1, 2, 3, 4$

5.2.1), we calculate new task attributes (see Sections 5.2.2 and 5.2.3) and the client utility in the case of dependent tasks (see Section 5.2.4).

### 5.2.1 Task Description

Compared to Chapter 4, in this model a task, $i$, can be not just interrupted due to the end of allocation period or a resource failure, but can also be stopped due to a lack of input data from other task(s). In this work, we define two roles for the tasks: to receive data (recipient-task) and to send data (sender-task). Some of the tasks can be sender-tasks and recipient-tasks at the same time. When a task is stopped, it means that resources can still be used by this task in case it can resume its work. If a task is interrupted, it means that it does not have any resources to be executed. In this model, a task can also be run with an inaccurate input data, when some of its sending tasks are stopped or interrupted. In this case, a task considers the last received data from the currently interrupted or stopped tasks in its calculations, which is potentially inaccurate in terms of the possible changes of a parameter estimated by this task. A task is considered to run with accurate input data only when it receives up-to-date results from all sending tasks, and those sending tasks are receiving accurate input data (if applicable), etc. In our previous work, a task can only be either running or idle. Here, task execution has specific conditions for running, and a task idling does

not always mean loss of resources for this task. Therefore, we introduce a *status* of task $Status_i\,(t)$ as in the following notation.

*Notation* 5.1. The status $Status_i\,(t)$ of task $i \in \mathbb{N}$ denotes a state of task execution at time $t \in \mathbb{R}$, i.e. whether a task is running with an accurate or inaccurate input data, or whether a task is stopped or interrupted.

$Status_i\,(t)$ can be assigned one of the following labels:

- 'interrupted' which denotes an interrupted task;
- 'stopped' which denotes a stopped task;
- 'inaccurate' which denotes a running task with inaccurate input data (if applicable);
- 'accurate' which denotes a running task with accurate input data.

Some tasks cannot have status 'inaccurate', because they do not need any input data from other tasks to be run. Therefore, if they are not interrupted or stopped, their status is always equal to 'accurate'.

Figure 5.1 shows the inter-dependencies among the tasks, depicted as a task tree. Here, each node denotes a task (e.g. $i = 1$), while each edge indicates a data-related interconnection between data sender $i$ and data recipient $j$ tasks with a weight $\alpha_{i,j}$. The weights define the level of relevance of the data from the sender-task $i$ in respect of the recipient-task $j$. The smaller this weight, the less impact task, $i$, has on the work of task, $j$. We assume that the sum of these weights for all sender-tasks which are connected directly to the same recipient-task is equal to 1.0 (e.g. $\alpha_{1,3} + \alpha_{2,3} = 1.0$).

*Definition* 5.1. A tree of tasks $Tr = \{(i,j)\,|\alpha_{i,j} \in [0,1]\}$ is a set of pairs $(i,j)$, of sender, $i$, and recipient, $j$, tasks with the level of relevance $\alpha_{i,j}$ of the task's $i$ data in respect of the task $j$. A sub-tree of tasks $sTr_k \in Tr$ is a sub-set of $Tr$ with the task $k$ as a root task of this sub-tree, where $i, j, k \in \mathbb{N}$.

We assume that each task (except the root task) sends its data to the corresponding recipient-task, and each sender-task has only one recipient-task for simplicity. However, every recipient-task might have one or more sender-task(s). The final result (e.g. an average temperature in the building) is produced by the *root task* at the top of a tree. The *leaf-tasks* are tasks which do not have their corresponding sender-task(s) (e.g. the

tasks "1" and "2" in Figure 5.1). This model of task inter-dependencies can follow, for example, a scenario of data aggregation by the root-task such as an analysis of complex climate parameters in the building, etc.

Every task has its own layer $Layer_i \in \mathbb{N}$ in a task tree, where the lowest-layer, $Layer_i = 0$, tasks are leaf-tasks and the highest-layer, $Layer_i = N_{lay} - 1$ where $N_{lay} \in \mathbb{N}$ is the number of layers in a tree, task is the root task. This layer is considered as one of the task's private attributes. The upper-layer tasks are considered of a higher importance to the client than the lower-layer tasks, because they are responsible for processing the data from a larger sub-tree of tasks. Therefore, their interruption will lead to a larger number of stopped tasks (all lower-layer tasks in a sub-tree) and, as a result, to the greater negative impact on the client utility, because the stopped tasks are considered as interrupted tasks in terms of the client utility as long as they do not produce the results. The upper-layer tasks are also considered to be more important to a client, because they perform more sophisticated calculations, which might affect the larger part of the client's system in the case of the prolonged interruptions. For example, it might affect one campus of a building if an upper-layer task is stopped, or one room in a building if a lower-layer task is stopped.

#### 5.2.1.1 Delay Times and Damping Times

A delay time, as discussed above, defines for how long a task can be running when it has to use inaccurate data for its calculations and it stops after this time. A *damping time* defines for how long a task can be not running without substantial negative consequences in terms of some uncontrollable parameter (e.g. a significant rise in the temperature). To present those times in a formal way, we have to define a parameter, which is estimated by a client task.

*Definition* 5.2. An estimated parameter $P_{i,S_i}(t) \in \mathbb{R}$ for task $i \in \mathbb{N}$ with the corresponding set of direct sender-tasks $S_i = \{m, ..., k\}$ at time $t$ is some real-life characteristic, which is continuous or can be presented as continuous over time $t \in \mathbb{R}$, considering $\left(P_{i,S_i}(t + \Delta t) - P_{i,S_i}(t)\right) / P_{i,S_i}(t) \ll 1$ and $\Delta t \in \mathbb{R}$ is an arbitrary small time step.

This parameter $P_{i,S_i}(t)$ is estimated directly by task $i$, if this task belongs to the lowest layer of a tree, i.e. $S_i \in \emptyset$. If task $i$ belongs to any upper layer of a tree, i.e. $S_i \notin \emptyset$, then $P_{i,S_i}(t)$ is estimated as a linear combination of all parameters $P_{j \neq i,S_j}(t)$ sent by

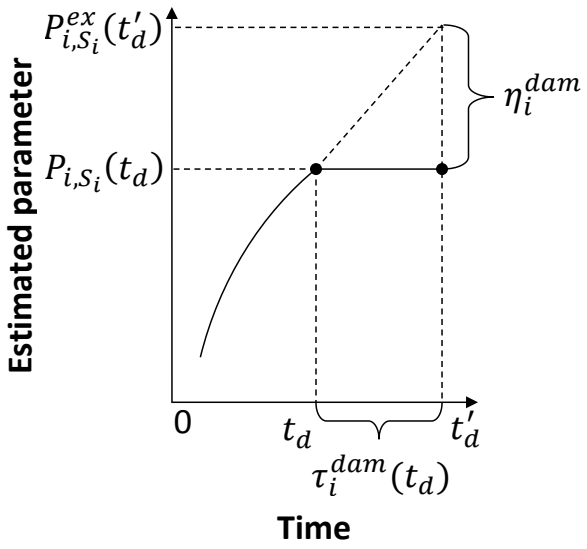



FIGURE 5.2: The calculation of $\tau_i^{dam}(t_d)$ when task $i$ is interrupted

its sender-task(s) as presented in the following equation:

$$P_{i,S_i}(t) = \sum_{j \in S_i} \alpha_{j,i} \times P_{j \neq i,S_j}(t), \; S_i \notin \emptyset, \; S_j \in \emptyset \; or \; S_j \neq \emptyset, \tag{5.1}$$

where $i, j \in \mathbb{N}$. If all tasks run without interruptions, then each task sends and receives (if applicable) the corresponding parameters in time. However, if any task has been interrupted, then it does not estimate this parameter any more. In this way, a parameter is assumed to change in a way, which is not controlled by this task. As a result, the longer this task is not running, the higher probability that the change of this parameter might have a substantial negative effect for a client. We assume that this effect occurs when a damping time is passed since a task has been interrupted or stopped.

*Definition* 5.3. A damping time $\tau_i^{dam}(t_d) > 0$, starting at time $t_d \in \mathbb{R}$, for task $i \in \mathbb{N}$ is a duration of time at the end of which the last value of a parameter $P_{i,S_i}(t_d)$, estimated by this task before interruption, becomes so different from an extrapolated value of this parameter $P^{ex}_{i,S_i}\left(t'_d\right)$ at time $t'_d \in \mathbb{R}$ that the client considers the absolute difference $\Delta P_{i,S_i}\left(t'_d\right) = P^{ex}_{i,S_i}\left(t'_d\right) - P_{i,S_i}\left(t'_d\right)$, $(P_{i,S_i}\left(t'_d\right) = P_{i,S_i}(t_d))$, to be so large that it has likely affected its system in a noticeable way (e.g. the temperature level has fallen below 0° C, compared to a normal room temperature).

It has to be noted that the damping time $\tau_i^{dam}(t_d)$ does not change during the interruption period $\tau_{i,l}^{int}$, where $t_d$ indicates the start of damping time and $l$ indicates the counter of interruptions. However, a damping time might be different for another interruption period because of the various reasons e.g., the speed of change in an estimated parameter. The other possible reason is that an interrupted task has been run with an inaccurate input data before time $t_d$. If this case, the damping time will be shorter than normal, because this task did not properly control an estimated parameter before $t_d$. Therefore, an error in its commands towards actuators has already made client's system a bit closer to a situation when the uncontrollable changes of the parameter affect noticeably this system (e.g. a climate in a building).

*Notation* 5.2. The value of $\eta_i^{dam} \in \mathbb{R}$ indicates when a difference $\Delta P_{i,S_i}(t)$ becomes so large that the client's system has been noticeably affected, making the end of the damping time, where $i \in \mathbb{N}$ and $t \in \mathbb{R}$. This value is chosen externally by the owner of the tasks, according to the nature of an estimated parameter (e.g. the difference in 10° C from a normal temperature e.g., 20° C, in an office room will be definitely uncomfortable for people).

A process of calculation of the damping time $\tau_i^{dam}(t_d)$ is depicted in Figure 5.2, where a linear extrapolation is chosen to present an estimation of $P_{i,S_i}^{ex}\left(t_d^{'}\right)$, and time $t_d$ is the beginning of an interruption period, while time $t_d^{'}$ is the end of damping time. However, an interruption period may continue after time $t_d^{'}$, but it means a significant decrease in the client utility (i.e. its utility is asymptotically close to zero). Now, we define a delay time, according to our previous discussion.

*Definition* 5.4. A delay time $\tau_i^{del}(t_y, t) > 0$, starting at time $t_y \in \mathbb{R}$, for recipient-task $i \in \mathbb{N}$ is the duration of time when this task can still run, but it has to use an inaccurate input data for its calculations due to an interruption of some sender-task(s) $Int = \{j, ..., k\}$, where $Int \in sTr_i$, $j, k \neq i$, and at the end of this time the absolute difference $\Delta P_{i,S_i}\left(t_y^{'}\right)$ between the last result $P_{i,S_i}(t_y)$ produced by task $i$ with an accurate input data and its extrapolated value $P_{i,S_i}^{ex}\left(t_y^{'}\right)$ at time $t_y^{'}$ becomes so large that a client decides to stop this task.

The end of a delay time for task $i$ denotes that this task cannot be run due to a significant error in its results as a result of usage of an inaccurate input data for a long time. However, the end of this time does not mean that the lack of control of some parameter has affected noticeably client's system, compared to a damping time. It is intuitively understandable that if a task is running even with an inaccurate input data,

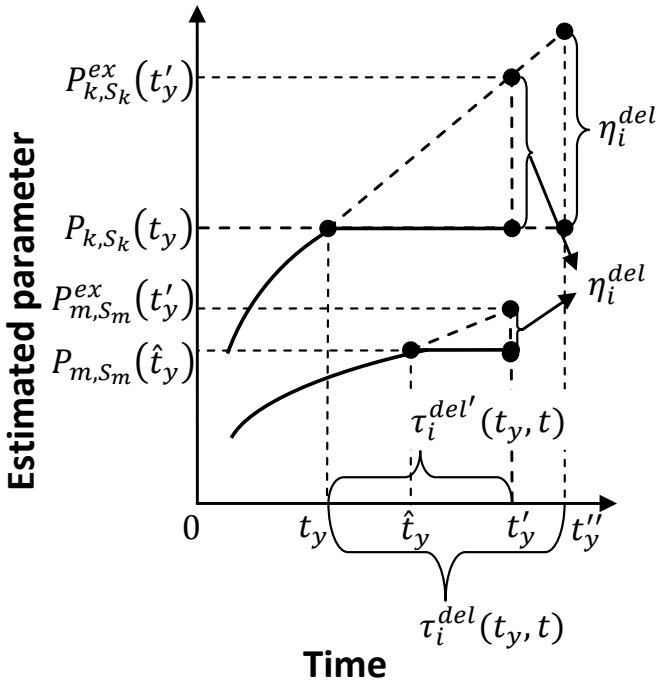



FIGURE 5.3: A delay time estimation when two sender-tasks are interrupted

it still controls an estimated parameter to some extent, while if a task is not running, it does not control this parameter at all. Basically, stopping of recipient-task at the end of delay time just means that this task is unable to control an estimated parameter even to some extent, because of a significant error in its calculations.

*Notation* 5.3. The value of $\eta_i^{del} \in \mathbb{R}$ indicates when a difference $\Delta P_{i,S_i}(t)$ becomes so large that it means the task cannot run without its results being unacceptably inaccurate, making the end of the delay time, where $i \in \mathbb{N}$ and $t \in \mathbb{R}$. This value is chosen externally by the owner of the tasks, according to the nature of an estimated parameter (e.g. the difference in 1° C from any temperature level e.g., 23° C, in an office room might be considered as a significant error in task's estimation of temperature).

It has to be noted that a delay time $\tau_i^{del}(t_y, t)$, compared to a damping time $\tau_i^{dam}(t_d)$, may change over time, while a damping time may change only at the beginning of the next interruption period. Therefore, a damping time $\tau_i^{dam}(t_d)$ only depends on its start time $t_d$, while a delay time $\tau_i^{del}(t_y, t)$ also depends on the current time $t$. This distinction can be explained, if we describe how $\Delta P_{i,S_i}(t)$ is calculated for both times. If an interrupted (stopped) task $i$ belongs to the lowest layer of a task tree, then

$\Delta P_{i,S_i}(t) = P^{ex}_{i,S_i}(t) - P_{i,S_i}(t)$, where $S_i \in \emptyset$. However, if a task belongs to any upper layer, $\Delta P_{i,S_i}(t)$ is calculated as a linear combination of all $\Delta P_{j \neq i,S_j}(t)$, $j \in S_i$ from the corresponding sender-task(s), which are stopped, interrupted or run with an inaccurate input data. Consequently, these sender-tasks can have their own sender-task(s), which are not run appropriately, etc. as presented in the following equations for time $t$:

$$\begin{aligned} &\Delta P_{i,S_i}(t) = \sum_{j \in S_i} \alpha_{j,i} \Delta P_{j \neq i,S_j}(t), \ S_i \neq \emptyset, \ S_j \in \emptyset. \\ &\qquad or \\ &\Delta P_{i,S_i}(t) = \sum_{j \in S_i} \alpha_{j,i} \sum_{k \in S_j} \alpha_{k,j} \Delta P_{k \neq j,S_k}(t), \ S_i \neq \emptyset, \ S_j \neq \emptyset, \ S_k \in \emptyset. \end{aligned} \tag{5.2}$$

If a task $i$ has been stopped or interrupted, all lower layer tasks which send their data directly or indirectly to this task are also stopped because they have nowhere to send their results, i.e. their statuses will be either 'interrupted' or 'stopped'. Consequently, when a client estimates a damping time for task $i$, $\Delta P_{i,S_i}(t)$ consists of all tasks from the lower layers, which belong to its sub-tree, i.e. $j \in sTr_i$. The lower layer tasks cannot be run if the upper layer task is stopped and, therefore, their status does not change till task $i$ is run. In this way, a damping time does not change over time as well, except for the next interruption period. It also has to be noted that a value of $\Delta P_{i,S_i}(t)$ increases faster for a damping time than for a delay time, because all lower layer tasks are contributing into increase of this value.

In the case of a delay time, the lower layer tasks which statuses are 'interrupted', 'stopped' or 'inaccurate' increase a value of $\Delta P_{i,S_i}(t)$ over time, but tasks with status 'inaccurate' does not increase it as fast as tasks with statuses 'interrupted' or 'stopped'. In this way, $\Delta P_{i,S_i}(t)$ increases more slowly for the delay time, and it may change if some of the lower layer tasks change status. For example, if one of the interrupted sender-task(s) obtains resources and runs with accurate input data before its recipient-task is stopped, then a delay time is extended for this recipient-task, because this sender-task does not contribute into increase of $\Delta P_{i,S_i}(t)$ any more. If one more sender-task is stopped, interrupted or run with an inaccurate input data, then a delay time decreases for task $i$, because this sender-task starts contributing into an error $\Delta P_{i,S_i}(t)$ of parameter estimation. In this way, the delay time can be re-calculated, depending on the change in statuses of the lower layer tasks.

For instance, Figure 5.3 shows two interrupted (stopped) sender-tasks with the identifiers $k$ and $m$, which contribute into a delay time for a recipient-task with an identifier $i$. Consequently, task $k$ has been interrupted at time $t_y$, and task $m$ has been interrupted at time $\hat{t}_y$, where $t_y < \hat{t}_y$. When task $k$ has been interrupted, an estimated delay time was $\tau_i^{del}(t_y, t)$, and only $\Delta P_{k,S_k}(t)$, $k \in S_i$ has been contributing into increase of an error $\Delta P_{i,S_i}(t)$ of parameter estimation for task $i$. However, when task $m$ has been interrupted, the delay time became shorter and equal to $\tau_i^{del'}(t_y, t)$, because both $\Delta P_{k,S_k}(t)$ and $\Delta P_{m,S_m}(t)$, $k, m \in S_i$ contributed to the increase of the error $\Delta P_{i,S_i}(t)$. It is also important to mention that if task $k$ is run with accurate input data, the start time for the task's $i$ delay time will change from $t_y$ to $\hat{t}_y$, because task $k$ does not contribute into a delay time any more. The details of how the delay and damping times are calculated is explained in the following sections.

#### 5.2.1.2 Task Attributes

In this chapter, the number of private attributes increases because of the dependencies among tasks. The task's private attributes, described in Chapter 4, are not constants chosen by a client before a task is launched in this chapter, but they are variables which depend on the damping time. Some new private attributes are also variables which depend on the delay time.

In particular, the pair of task attributes $\left(\tau_{int[i]}^{max}(t_d), \epsilon_{int[i]}(t_d)\right)$ (see Definitions 4.4, 4.2) determines how quickly the possible increment in the client utility for the following allocation period decreases towards zero during an interruption period. As long as those attributes are concerned with the duration of interruption, they should reflect impact at the end of damping time on the client's utility, i.e. the client utility has to decrease substantially after this time ends. Therefore, these attributes depend on the start moment of time $t_d$ of damping time $\tau_i^{dam}(t_d)$ for task $i$. In this way, the value $\tau_{int[i]}^{max}(t_d)$, which shows the duration of interruption after which (if not earlier due to other factors such as a total interruption) a possible increment in the client utility decreases in more than two times (see Equation (4.7)), can be calculated as in the following equation:

$$\tau_{int[i]}^{max}(t_d) = \tau_i^{dam}(t_d) - \tau^*, \tag{5.3}$$

where $\tau^* < \tau_i^{dam}(t_d)$, and $\tau^* \in \mathbb{R}$ is chosen by a client. In this way, the end of $\tau_{int[i]}^{max}(t_d)$ may reflect the end of a damping time. The choice of $\tau^*$ reflects how substantial will be damage to the client's system at the end of a damping time e.g., if a damping time

ends after $\tau^{max}_{int[i]}(t_d)$ where the utility will be close to zero. Consequently, $\epsilon_{int[i]}(t_d)$ is calculated proportionally to $\tau^{max}_{int[i]}(t_d)$ as in the following equation:

$$\epsilon_{int[i]}(t_d) = k \times \tau^{max}_{int[i]}(t_d), \tag{5.4}$$

where $k \in [0,1]$, $k \in \mathbb{R}$. The other pair of task attributes $\left(\tau^{max}_{tot[i]}(t_d), \epsilon_{tot[i]}(t_d)\right)$ (see Definitions 4.6, 4.3) determines how quickly the possible increment in the client utility for the following allocation period decreases towards zero, considering a duration of total interruption from the beginning of task execution $t^0$. As long as those attributes also reflect the impact of interruption on the client's utility, we calculate them in proportion to $\left(\tau^{max}_{int[i]}(t_d), \epsilon_{int[i]}(t_d)\right)$, and they show when a total interruption starts noticeably affecting client's system. For example, if there are frequent uncontrolled changes in the temperature in a greenhouse due to task interruptions, their total impact on the plants may result, for example, in the change of their blooming time even if each single interruption had not exceeded its corresponding damping time. That is,

$$\begin{aligned} \tau^{max}_{tot[i]}(t_d) &= m \times \tau^{max}_{int[i]}(t_d), \\ \epsilon_{tot[i]}(t_d) &= l \times \epsilon_{int[i]}(t_d), \end{aligned} \tag{5.5}$$

where $m, l \in \mathbb{R}$. In this model, the client utility for task $i$ is not only affected by the durations of single and total interruptions as in Chapter 4, but it is also affected by the level of accuracy of input data and, as a result, by a duration of delay time $\tau^{del}_i(t_y, t)$. The two additional attributes $\left(\tau^{max}_{del[i]}(t_y, t), \epsilon_{del[i]}(t_y, t)\right)$ determine how quickly the current increment in the client's utility (the utility keeps increasing for a task which runs with an inaccurate input data) decreases in respect of the end of a delay time. In other words,

*Definition* 5.5. The attribute $\tau^{max}_{del[i]}(t_y, t) > 0$ is the duration of time when task $i \in \mathbb{N}$ is running, but using inaccurate input data for so long that the increment in the task's utility decreases in a half at the end of this time, where $t_y, t \in \mathbb{R}$.

*Notation* 5.4. The attribute $\epsilon_{del[i]}(t_y, t) > 0$ determines the speed of decrease in the corresponding increment of the client's utility, which increases in the approximate proximity of the end of $\tau^{max}_{del[i]}(t_y, t)$, where $i \in \mathbb{N}$, $t_y, t \in \mathbb{R}$.

The value of $\tau^{max}_{del[i]}(t_y, t)$ depends on the delay time at time $t$, because the closer task to the end of delay time, the more substantially it should affect the client utility.

$$\tau^{max}_{del[i]}(t_y, t) = \tau^{del}_i(t_y, t) + \tau^{**}, \tag{5.6}$$

where $\tau^{**} \in \mathbb{R}$ is a duration of time chosen by a client in order to determine how fast an inaccurate input data will affect the client utility towards the end of delay time. Consequently, the longer $\tau^{max}_{del[i]}(t_y, t)$ is compared to $\tau^{del}_{i}(t_y, t)$, the less inaccurate input data affects the client utility towards the end of delay time. In contrast with the damping time, we assume that the client utility is not expected to be close to zero at the end of the delay time and, therefore, the end of delay time can be either equal to $\tau^{max}_{del[i]}(t_y, t)$ or ends before $\tau^{max}_{del[i]}(t_y, t)$ (i.e. before a significant decrease in the increment of the client utility). Consequently, a value of $\epsilon_{del[i]}(t_y, t)$ is calculated as a proportion of $\tau^{max}_{del[i]}(t_y, t)$ as in the following equation:

$$\epsilon_{del[i]}(t_y, t) = d \times \tau^{max}_{del[i]}(t_y, t), \tag{5.7}$$

where $d \in [0, 1]$, $d \in \mathbb{R}$. Now, we describe other attributes which determine the inter-dependencies among tasks inside a task tree. Those attributes have already been mentioned in this chapter and they are the layer $Layer_i$ of a tree to which task $i$ belongs, the status $Status_i(t)$ of its execution and the level of relevance $\alpha_{i,j}$ of its data in respect of recipient-task $j$ (except for the root task). Finally, the modified old attributes $OldAttrib_{i,t_d}$ (compared to Chapter 4) and additional new attributes $NewAttrib_{i,j,t_y,t}$ are presented below.

$$\begin{aligned} OldAttrib_{i,t_d} &= \left(\tau^{max}_{int[i]}(t_d), \epsilon_{int[i]}(t_d), \tau^{max}_{tot[i]}(t_d), \epsilon_{tot[i]}(t_d), t^{exec}_{dl}, t^{c}_{dl}\right), \\ NewAttrib_{i,j,t_y,t} &= \left(\tau^{max}_{del[i]}(t_y, t), \epsilon_{del[i]}(t_y, t), Layer_i, Status_i(t), (j, \alpha_{i,j})\right), \end{aligned} \tag{5.8}$$

where $t^{exec}_{dl}$ and $t^{c}_{dl}$ are the non-modified old attributes. Here, $t^{exec}_{dl}$ defines the moment of time when the task's execution period ends, while $t^{c}_{dl}$ defines a nominal deadline of negotiation. A notation $(j, \alpha_{i,j})$ defines a dependence between sender-task $i$ and recipient-task $j$ in terms of the data exchange. Finally, the modified tuple of task's private attributes is presented in the following equation:

$$Task_{i,j,t_y,t_d,t} = (OldAttrib_{i,t_d}, NewAttrib_{i,j,t_y,t}). \tag{5.9}$$

As in previous chapter, the specification $Spec_{i,l}$ (see Formula (4.2)) of task $i$, which is available to the GRA, includes the task's identifier $i$, and the minimum acceptable $\tau^{min}_{i,l}$ and maximum $\tau^{max}_{i,l}$ durations of execution (see Definitions 4.7, 4.8) used in negotiation with the GRA during the interruption period $\tau^{int}_{i,l}$.

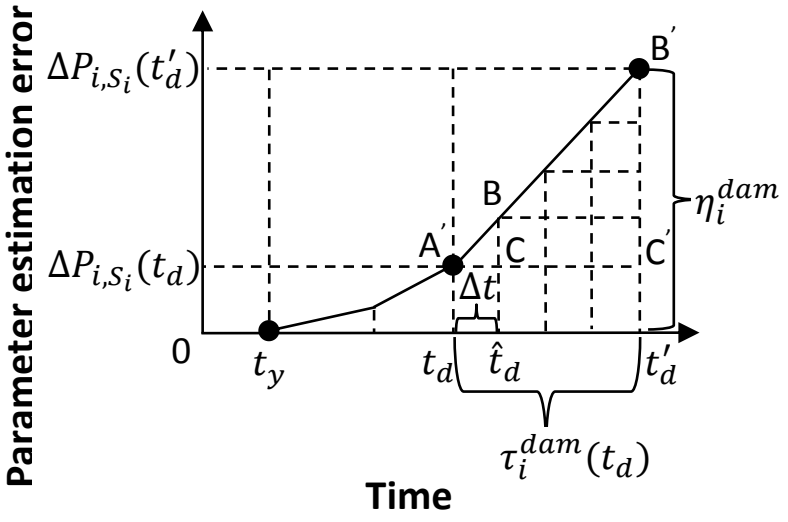



FIGURE 5.4: A calculation of damping time for the $i^{th}$ task

### 5.2.2 Calculation of Damping Time

The damping time is calculated by the client in order to determine when an interruption of a particular task affects significantly its system. When a task is interrupted, it does not estimate and control any parameter(s) (e.g. the temperature level), which may then change unpredictably and significantly during this interruption. The longer this interruption, the larger difference between the task's last estimation of this parameter before interruption and an extrapolated value of this parameter at time as discussed in the previous section.

Figure 5.4 shows an increase in the difference $\Delta P_{i,S_i}(t)$ over time $t$, starting at time $t_y$ till time $t_d^{'}$. This figure depicts one of the possible scenarios when a damping time has to be calculated. That is, a task is assumed to run with an accurate input data ($Status_i$ = 'accurate') before time $t_y$, then it starts running with an inaccurate input data ($Status_i$ = 'inaccurate') till time $t_d$, increasing the difference $\Delta P_{i,S_i}(t)$. At time $t_d$ a task stops ($Status_i$ = 'stopped') due to the end of the delay time, and an interruption period starts. In this way, a damping time has to be calculated after time $t_d$. As depicted in this figure, the difference $\Delta P_{i,S_i}(t_d) = P^{ex}_{i,S_i}(t_d) - P_{i,S_i}(t_d)$ ($P_{i,S_i}(t_d) = P_{i,S_i}(t_y)$) has already increased before time $t_d$, and this situation has to be considered. In this case, a difference $\Delta P_{i,S_i}(t_d) = \eta_i^{del}$ reflects the end of the delay time. If a task was run with an accurate input data before time $t_d$ and then it was interrupted due to a loss of

resources, then $\Delta P_{i,S_i}(t_d) = 0$. If a task was run with an inaccurate input data before time $t_d$ and then it was interrupted due to a loss of resources before $\eta_i^{del}$ is reached, then $\Delta P_{i,S_i}(t_d) < \eta_i^{del}$.

Considering a linear extrapolation of $P_{i,S_i}(t)$ over time, we can calculate a damping time $\tau_i^{dam}(t_d)$ at some point in time $\hat{t}_d$ after time $t_d$ for task $i$, according to the triangles' similarity such as the triangles $A^{'}BC$ and $A^{'}B^{'}C^{'}$. Here, our assumption is that an extrapolation of an estimated parameter is linear, and it is considered for simplicity in our modelling. The damping time can be calculated at any time after $t_d$ e.g., in our model, it is calculated in one conventional second after a task has been interrupted or stopped. Assume the damping time is calculated at time $\hat{t}_d$ after $t_d$, and $\Delta t = \hat{t}_d - t_d$. According to the triangles $A^{'}BC$ and $A^{'}B^{'}C^{'}$ similarity (see Figure 5.4), where $A^{'}C^{'} = \tau_i^{dam}(t_d)$, $B^{'}C^{'} = \eta_i^{dam} - \Delta P_{i,S_i}(t_d)$, $BC = \Delta P_{i,S_i}\left(\hat{t}_d\right) - \Delta P_{i,S_i}\left(\hat{t}_d - \Delta t\right)$ and $A^{'}C = \Delta t$, the damping time $\tau_i^{dam}(t_d)$ is calculated as in the following equation:

$$\tau_i^{dam}(t_d) = \frac{\eta_i^{dam} - \Delta P_{i,S_i}(t_d)}{\Delta P_{i,S_i}\left(\hat{t}_d\right) - \Delta P_{i,S_i}\left(\hat{t}_d - \Delta t\right)} \times \Delta t. \tag{5.10}$$

As we discussed above, a damping time ends when the absolute difference $\Delta P_{i,S_i}(t)$ rises to a value $\eta_i^{dam}$, chosen by a client, according to the nature of an estimated parameter.

### 5.2.3 Calculation of Delay Time

The delay time is calculated by the client in order to determine when a recipient-task has to stop due to a significant error in its estimation of a parameter (e.g. the temperature level). This error occurs as a result of inaccurate input data, which a recipient-task has to use when it does not receive an up-to-date data from some of its sender-task(s). When some of its sender-task(s) are stopped, interrupted or run with an inaccurate input data, a recipient-task does not produce the accurate results as well. A speed of increase in the difference $\Delta P_{i,S_i}(t)$ for a recipient-task, $i$, over time depends on how many sender-task(s) are contributing into it and how much relevant their data for this recipient-task's estimations as we discussed above in this chapter.

Figure 5.5 shows a change in the difference $\Delta P_{i,S_i}(t) = P_{i,S_i}^{ex}(t) - P_{i,S_i}(t)$ over time, starting at time $t_y$ till time $t_y^{'}$, for task $i$ which is a recipient-task. This figure depicts only one scenario of how a delay time may change over time. In particular, we assume

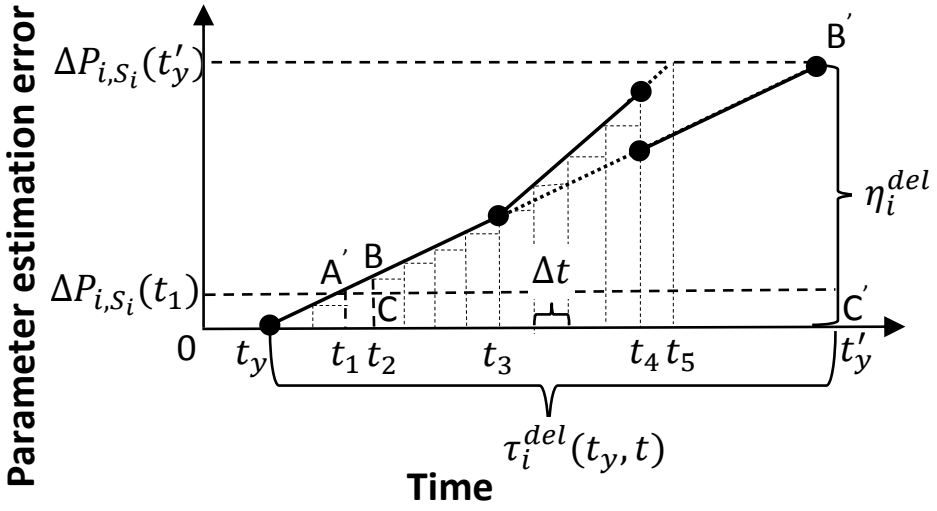



FIGURE 5.5: A calculation of delay time $\tau_i^{del}(t_y, t)$ for task $i$

that task $i$ has $Status_i =$ 'accurate' before time $t_y$ (i.e. $\Delta P_{i,S_i}(t) = 0$), then it changes to $Status_i =$ 'inaccurate' due to some sender-task(s), switching to $Status_j$={'interrupted', 'stopped', 'inaccurate'} from $Status_j =$ 'accurate', $j \neq i$, $j \in S_i$. At time $t_3$, some more sender-task(s) switch to $Status_j$={'interrupted', 'stopped', 'inaccurate'} and, therefore, a speed of increase in the difference $\Delta P_{i,S_i}(t)$ rises. At time $t_4$, some sender-task(s) switch to $Status_j =$ 'accurate' and, hence, a speed of increase in the difference $\Delta P_{i,S_i}(t)$ drops down. A delay time $\tau_i^{del}(t_y, t)$ ends when the difference $\Delta P_{i,S_i}\left(t'_y\right) = \eta_i^{del}$. As long as a delay time depends on how many sender-task(s) are running with accurate data, a client has to calculate this time frequently for each recipient-task with some small time step $\Delta t$ (e.g. one conventional second) in order to notice status changes of the corresponding sender-task(s) in time. For example, in Figure 5.5, a delay time $\tau_i^{del}(t_y, t)$, calculated at any time point $t$ in the interval $]t_y, t_3]$, is equal to $t'_y - t_y$, while in the interval $]t_3, t_4]$ it is equal to $t_5 - t_y$.

A delay time $\tau_i^{del}(t_y, t_2)$ is calculated, considering a similarity of triangles such as the triangles $A'BC$ and $A'B'C'$ in Figure 5.5 at time $t_2$. Considering $A'C = \Delta t$, $A'C' = \tau_i^{del}(t_y, t_2) - (t_2 - t_y - \Delta t)$, $BC = \Delta P_{i,S_i}(t_2) - \Delta P_{i,S_i}(t_2 - \Delta t)$ and $B'C' = \eta_i^{del} - \Delta P_{i,S_i}(t_2 - \Delta t)$, a delay time at time $t_2$ is calculated as in the following equation:

$$\tau_i^{del}(t_y, t_2) = t_2 - t_y - \Delta t + \frac{\eta_i^{del} - \Delta P_{i,S_i}(t_2 - \Delta t)}{\Delta P_{i,S_i}(t_2) - \Delta P_{i,S_i}(t_2 - \Delta t)} \times \Delta t. \tag{5.11}$$

### 5.2.4 Client Utility

We assume that each client has to run all its tasks continuously and simultaneously as discussed above. It is desirable for a client that every task runs without interruptions, and if an interruption has occurred its duration should be as short as possible. A client attempts to reduce not just the duration of each single interruption, but also a total duration of interruptions which affects an effectiveness of task execution. In other words, every task $i$ has its own utility $U_{Cl}^{i}$ (see Chapter 4) which reflects a continuity of its execution for a length of time $\tau_{dl}^{exec}$ and this time is considered to be the same for all tasks in one job as well as the tasks' deadline of execution $t_{dl}^{exec}$. We believe that as long as all tasks are implicitly or explicitly connected in a tree, then they represent one system. Therefore, all client tasks have the same deadline, and preferably they have to be run simultaneously, but a delay time is still allowed and estimated, considering the dependencies among tasks.

The delay time means the duration of time when the utility of task $U_{Cl}^{i}$ is affected by inaccurate input data from sender-task(s). In this case, an effectiveness of task execution, which shows a success of task execution over time by increasing during allocation periods, will decrease to some extent during task run-time. Therefore, the effectiveness function, presented in Chapter 4, has to be modified by embedding a new damping function $D(\cdot)$ for the allocation period, which reduces the speed of increase in the client's utility during an allocation period.

*Definition* 5.6. The value of $\tau_{cdel[i]}(t_y, t) > 0$ at time $t \in \mathbb{R}$ is the duration of time during which task $i \in \mathbb{N}$ is running with an inaccurate input data, starting from time $t_y \in \mathbb{R}$.

*Notation* 5.5. The damping function $D\left(\tau_{cdel[i]}(t_y, t)\right) \in [0, 1]$ at time $t \in \mathbb{R}$ shows how much the speed of increase in the effectiveness of task execution during allocation period has to be reduced, when the task has been running with an inaccurate input data for the duration of time $\tau_{cdel[i]}(t_y, t)$, where $i \in \mathbb{N}$ and $t_y \in \mathbb{R}$.

The value of this damping function changes during $\tau_{cdel[i]}(t_y, t)$, and it is presented in the following equation:

$$D\left(\tau_{cdel[i]}(t_y, t)\right) = \frac{1}{e^{\left(\tau_{cdel[i]}(t_y,t) - \tau_{del[i]}^{max}(t_y,t)\right)/\epsilon_{del[i]}(t_y,t)} + 1}. \tag{5.12}$$

A part of the effectiveness function $E(\cdot)$ (see Equation (4.9)), which has not been modified, depicts a linear increase during the allocation period $\tau_{i,l}^{all}$ for task $i$. It is presented in the following equation as an estimate $Es(t, E(\cdot))$ at time $t$, which considers the start time $t_{i,l}^{str}$ of the allocation period $\tau_{i,l}^{all}$ and an effectiveness of task execution at the end $t_{i,l-1}^{end}$ of the allocation period $\tau_{i,l-1}^{all}$ .

$$Es\left(t, E(t_{i,l-1}^{end})\right) = \frac{\left(1 - E\left(t_{i,l-1}^{end}\right)\right) \times t + E\left(t_{i,l-1}^{end}\right) \times t_{dl}^{exec} - t_{i,l}^{str}}{t_{dl}^{exec} - t_{i,l}^{str}}. \tag{5.13}$$

A modified part of the effectiveness function $E(\cdot)$, depicts an additional coefficient, produced by $D\left(\tau_{cdel[i]}(t_y, t)\right)$, when a task is running with inaccurate input data. It also has to be noted that an old effectiveness function $E(\cdot)$ is substituted with a new modified effectiveness function $\hat{E}(\cdot)$ in Equation (5.13). Consequently, the modified effectiveness function $\hat{E}(t)$ at time $t$ is described as in the following equation:

$$\hat{E}(t) = \begin{cases} Es\left(t, \hat{E}\left(t_{i,l-1}^{end}\right)\right) I\left(\tau_{i,l}^{int}\right) T\left(\tau_{i,l}^{tot}\right) D\left(\tau_{cdel[i]}(t_y, t)\right), \\ \qquad\qquad if\ \tau_{i,l}^{all} \neq 0\ and\ \tau_i^{del}(t_y, t) \neq 0, \\ Es\left(t, \hat{E}\left(t_{i,l-1}^{end}\right)\right) I\left(\tau_{i,l}^{int}\right) T\left(\tau_{i,l}^{tot}\right), \\ \qquad\qquad if\ \tau_{i,l}^{all} \neq 0\ and\ \tau_i^{del}(t_y, t) = 0, \\ \hat{E}\left(t_{i,l-1}^{end}\right), \\ \qquad\qquad if\ \tau_{i,l}^{all} = 0, \end{cases} \tag{5.14}$$

where $D\left(\tau_{cdel[i]}(t_y, t)\right)$ is a new component, compared to $E(t)$. The durations of time $\tau_{i,l}^{int}$ and $\tau_{i,l}^{tot}$ are the durations of a single and total interruption periods prior to the allocation period $\tau_{i,l}^{all}$ as defined in Chapter 4. It has to be noted that the values of the damping functions $I\left(\tau_{i,l}^{int}\right)$ and $T\left(\tau_{i,l}^{tot}\right)$ are the constants within an allocation period $\tau_{i,l}^{all}$, which reflect the utility loss due to these prior interruptions. However, $D\left(\tau_{cdel[i]}(t_y, t)\right)$ decreases towards zero within an allocation period (i.e. $\tau_{i,l}^{all} \neq 0$), if the recipient-task is using inaccurate data (i.e. $\tau_i^{del}(t_y, t) \neq 0$) from its sender-task(s). Consequently, there is no additional damping function $D(\cdot)$, if the recipient-task is running (i.e. $\tau_{i,l}^{all} \neq 0$) with accurate input data (i.e. $\tau_i^{del}(t_y, t) = 0$).

Compared to Chapter 4, an interruption of one task in this work may significantly affect not only its own utility $U_{Cl}^{i}$, but also the total utility of a group of tasks (with consideration of delay times) which are connected with this one by means of the data exchange. In the worst-case scenario, a prolonged interruption of one task may lead

to an interruption of the whole client system. It also has to be noted that $U^i_{Cl}$ is not calculated as the sum of trapeze squares, but as the square under the broken curve $\hat{E}(t)$, which is estimated numerically and normalised in the same way as in Chapter 4.

$$U^i_{Cl} = \frac{1}{S_{max}} \sum_{l=1}^{L_i} \int_{t^{str}_{i,l}}^{t^{end}_{i,l}} \hat{E}(t)\, dt, \tag{5.15}$$

where $S_{max}$ is the largest possible square under $\hat{E}(t)$, if a task runs without interruptions during $\tau^{exec}_{dl}$ (see Chapter 4), and $L_i$ is the total number of allocation periods within $\tau^{exec}_{dl}$. As in Chapter 4, the utility of each task $U^i_{Cl}$ does not change while they are inside an interruption period (see Equation (4.9) when $\tau^{all}_{i,l} = 0$), but the duration of interruption reduces its utility during the next allocation period (see Equation (4.9) when $\tau^{all}_{i,l} \neq 0$), compared to that one which could be in the case of a shorter interruption. In this work, we also consider that the task's utility decreases, while the dependent task is interrupted as presented by the damping function $D(\cdot)$.

We also assume that all tasks are relevant to the client, but with a different degree for each task. For instance, the temperature level in a single room can be less relevant to a client than the temperature level in the whole building. Therefore, we calculate the total utility of the client's system as a sum $U_{total}$ of all task utilities $U^i_{Cl}$ with the respective coefficients $\varpi_i$ which denote the level of relevance of this task in respect of the client's system (e.g. building). The sum of $\varpi_i$ over all client tasks is assumed to be equal to 1.0, where the total sum of all $\varpi_i$ from the same layer of the graph is equal for each layer. For example, if there are four layers in a graph, then the head task has the relevance coefficient $\varpi_i = 0.25$, the sum of $\varpi_i$ for the next layer tasks is equal to 0.25, etc. In this way, we ensure that our assumption about the higher importance of the upper-layer tasks (see Section 5.2.1) is implemented, and it is presented in Equation (5.16).

$$U_{total} = \sum_{i=1}^{N} \varpi_i \times U^i_{Cl}, \tag{5.16}$$

where $N$ denotes the total number of tasks in one job.

TABLE 5.1: List of notation for Section 5.2

| Symbol | Notation |
|---|---|

Continued on the next page

Continued from the previous page

| **Symbol** | **Notation** |
|---|---|
| $Status_i\,(t)$ | A status of execution of task $i \in \mathbb{N}$ at time $t \in \mathbb{R}$ e.g., 'interrupted'. |
| $\alpha_{i,j}$ | A coefficient which defines the level of importance of task's $i$ data for recipient-task $j$, where $\alpha_{i,j} \in [0,1]$ $i,j \in \mathbb{N}$ |
| $Tr$, $sTr_k$ | A tree and sub-tree of tasks respectively, where task $k$ is the root task in sub-tree $sTr_k$. Each node of this tree is a task and each edge denotes a data dependence between two tasks with a weight $\alpha_{i,j}$, where $i,j,k \in \mathbb{N}$. |
| $Layer_i$ | A layer of task tree $Tr$ to which task $i$ belongs to, where $i, Layer_i \in \mathbb{N}$. |
| $N_{lay}$ | The total number of layers in $Tr$, where $N_{lay} \in \mathbb{N}$. |
| $S_i$ | The set of sender-tasks in respect of a recipient-task $i$, where $i \in \mathbb{N}$. |
| $P_{i,S_i}\,(t)$ | A parameter (e.g. temperature level) at time $t$ which is estimated (monitored) by task $i$. If task $i$ does not belong to the lowest layer of $Tr$, then it is calculated as a linear combination of all parameters obtained from the corresponding sender-tasks $S_i$, where $P_{i,S_i}\,(t)\,, t \in \mathbb{R}$, $i \in \mathbb{N}$. |
| $P^{ex}_{i,S_i}\,(t)$ | An extrapolated value of parameter $P_{i,S_i}\,(t)$ at time $t$, where $P^{ex}_{i,S_i}\,(t)\,, t \in \mathbb{R}$, $i \in \mathbb{N}$. |
| $\tau_i^{dam}\,(t_d)$ | The damping duration of time, starting at $t_d$, at the end of which an interrupted (stopped) task $i$ significantly affects a controlled (monitored) parameter, where $\tau_i^{dam}\,(t_d) > 0$, $t_d \in \mathbb{R}$, $i \in \mathbb{N}$. |
| $\eta_i^{dam}$ | An absolute difference between the last produced value of parameter $P_{i,S_i}\,(t_d)$ by task $i$ before interruption and its extrapolated value $P^{ex}_{i,S_i}\,(t)$ at time $t$ $(t > t_d)$, which is considered to be substantial enough in order to conclude that a task's interruption has significantly affected this parameter, where $\eta_i^{dam}, t, t_d \in \mathbb{R}$, $i \in \mathbb{N}$. |

Continued on the next page

Continued from the previous page

| Symbol | Notation |
|---|---|
| $\tau_i^{del}(t_y, t)$ | The delay duration of time, starting at $t_y$, at the end of which task $i$ is stopped due to inability to produce the accurate enough results, because it has been receiving inaccurate input data for too long, where $\tau_i^{del}(t_y, t) > 0$, $t, t_y \in \mathbb{R}$, $i \in \mathbb{N}$. |
| $\eta_i^{del}$ | An absolute difference between the the last produced value of parameter $P_{i,S_i}(t_d)$ by task $i$ with accurate input data and its extrapolated value $P_{i,S_i}^{ex}(t)$ at time $t$ ($t > t_y$), which is considered to be substantial enough for task $i$ to be unable to produce any close to realistic results and, as a result, this task is stopped, where $\eta_i^{del}, t, t_y \in \mathbb{R}$, $i \in \mathbb{N}$. |
| $\tau_{int[i]}^{max}(t_d)$, $\epsilon_{int[i]}(t_d)$ | These variables have been defined in Section 4.2.1, and they generally reflect an influence of the duration of a single interruption on the client utility, where $\tau_{int[i]}^{max}(t_d) > 0$, $\epsilon_{int[i]}(t_d) > 0$, $t_d \in \mathbb{R}$, $i \in \mathbb{N}$. |
| $\tau_{tot[i]}^{max}(t_d)$, $\epsilon_{tot[i]}(t_d)$ | These variables also have been defined in Section 4.2.1, and they generally reflect an influence of the total duration of interruption on the client utility. In this chapter, the values of all these variables are determined to some extent by the damping time $\tau_i^{dam}(t_d)$ rather then empirically chosen, where $\tau_{tot[i]}^{max}(t_d) > 0$, $\epsilon_{tot[i]}(t_d) > 0$, $t_d \in \mathbb{R}$, $i \in \mathbb{N}$. |
| $\tau_{del[i]}^{max}(t_y, t)$ | The duration of time when task $i$ uses inaccurate input data to produce its results at the end of which the increment in the client's utility decreases in more than a half, compared to the case if it would use accurate input data. Hence, this variable is determined to some extent by the delay time $\tau_i^{del}(t_y, t)$, where $\tau_{del[i]}^{max}(t_y, t) > 0$, $t, t_y \in \mathbb{R}$, $i \in \mathbb{N}$. |
| $\epsilon_{del[i]}(t_y, t)$ | The value for task $i$ which determines the speed of the decrease in the increment of the client utility during a delay time $\tau_i^{del}(t_y, t)$, where $\epsilon_{del[i]}(t_y, t) > 0$, $t, t_y \in \mathbb{R}$, $i \in \mathbb{N}$. |
| $OldAttrib_{i,t_d}$ | A tuple of modified, compared to Chapter 4, private attributes for task $i$ such as $\tau_{int[i]}^{max}(t_d)$, $\epsilon_{int[i]}(t_d)$, $\tau_{tot[i]}^{max}(t_d)$, $\epsilon_{tot[i]}(t_d)$, where $t_d \in \mathbb{R}$, $i \in \mathbb{N}$. |

Continued on the next page

Continued from the previous page

| Symbol | Notation |
|---|---|
| $NewAttrib_{i,j,t_y,t}$ | A tuple of new, compared to Chapter 4, private attributes for task $i$ such as $\tau^{max}_{del[i]}(t_y,t)$, $\epsilon_{del[i]}(t_y,t)$, $Layer_i$, $Status_i(t)$ and $(j,\alpha_{i,j})$, where $t,t_y \in \mathbb{R}$, $i,j \in \mathbb{N}$. |
| $Task_{i,j,t_y,t_d,t}$ | An extended, compared to Chapter 4, tuple of task's private attributes. In this chapter, the private attributes include the modified old attributes $OldAttrib_{i,t_d}$ and new attributes $NewAttrib_{i,j,t_y,t}$, where $t,t_d,t_y \in \mathbb{R}$, $i,j \in \mathbb{N}$. |
| $\tau_{cdel[i]}(t_y,t)$ | The duration of time at time $t$ when task $i$ is running with inaccurate input data, starting from $t_y$, where $\tau_{cdel[i]}(t_y,t) > 0$, $t,t_y \in \mathbb{R}$, $i \in \mathbb{N}$. |
| $D\left(\tau_{cdel[i]}(t_y,t)\right)$ | A damping function which shows how much the duration of inaccurate task execution affects its effectiveness during the current allocation period, where $D\left(\tau_{cdel[i]}(t_y,t)\right) \in [0,1]$, $t,t_y \in \mathbb{R}$, $i \in \mathbb{N}$. |
| $\hat{E}(t)$ | A modified effectiveness function, compared to Chapter 4, which shows the success of task execution during an execution period $\tau^{exec}_{dl}$, considering its interruptions and executions with inaccurate input data, where $\hat{E}(t) \in [0,1]$, $t \in \mathbb{R}$. |
| $U_{total}$ | A total client utility for all tasks, considering each task's $i$ importance level $\varpi_i$ for a client, where $U_{total} \in [0,1]$, $\varpi_i \in [0,1]$, $i \in \mathbb{N}$. |

## 5.3 SimTask Re-allocation Strategy

Some research e.g., [54, 56, 57], focuses on execution of interdependent tasks, but it is generally lacking decision making mechanisms for a client in respect of allocation and execution of such tasks. In particular, it does not focus on how a client can avoid or shorten the delays and how those delays may affect a client system. Eventually, this might lead to an ineffective execution of data dependent tasks. The dependence among tasks is often depicted in terms of the data exchange and it has an explicit connection between a sender and recipient tasks. In other words, a dependent task can start running when the data is received from a corresponding sender-task. In our

model, the tasks do not have just an explicit dependence, but also an implicit one which means that a failure of the upper-layer tasks (data recipient) affect an execution of the corresponding lower-layer tasks (data sender) as much as the lower-layer tasks affect an execution of their upper-layer tasks. Considering implicit dependences allows us to be more realistic, because a Grid would not intend to waste its resources on a task whose execution is not useful and, therefore, this task has to be stopped until the upper-layer task will regain resources. We also take into account that tasks are not executed just once, but they have to be executed continually during a long period of time, which brings the problem of managing all failed tasks before their interruption significantly affects a controlled (monitored) parameter and other tasks.

In this way, we propose a new re-allocation strategy *SimTask* for a client which allows a client to exchange the allocated resources among its own tasks by negotiating with the Grid Resource Allocator (GRA). Consequently, the tasks which have been interrupted can continue their execution instead of other "fellow" tasks, and the tasks which have donated their resources to other tasks are called *donor-tasks*. The aim of this exchange is to avoid so long interruptions which cause a significant change in the controlled by the interrupted task parameter e.g., a significant drop in the temperature level due to its lack of control. As long as we consider only the length of time as a resource allocated by the GRA, then a task cannot share this resource with the other task, but it can give this resource to the other task.

A problem following from this decision is not only which task to stop in order to launch the interrupted one with a smaller loss in the client utility, but also whether the GRA is willing to make an exchange of the allocated resources between client tasks. A resource cost of this exchange can be represented by the computational resources which the GRA uses in order to make a decision, while it can be busy negotiating with other clients. A time cost for the GRA can be caused by a transition of a task from one physical resource to another one. In other words, the GRA is assumed to allow such exchanges, but only with a penalty. This penalty is that the GRA does not just exchange one of the client's interrupted tasks with another client's task following a client's request, but it negotiates with the client in respect of the remainder of time slot which has to be allocated to the interrupted task. That is, this remainder can be shortened after negotiation, if there is a large demand on resources.

This section describes decision making criteria for a client when it is necessary to re-allocate the interrupted task (see Section 5.3.1), discusses which donor candidate to

choose as a donor-task (see Section 5.3.2) and describes the whole algorithm of decision making for the re-allocation strategy (see Section 5.3.3).

### 5.3.1 Estimating the Criterion to Re-allocate an Interrupted Task

A client has to decide whether a situation is critical enough that an interrupted task should be donated resources from another client's task. The client aims to re-allocate resources to an interrupted task by using resources from another task only when its prolonged interruption threatens to decrease significantly client's utility. That is, in the case when a client cannot reach an agreement with the GRA for a long time e.g., the resources are scarce. Otherwise, the resource donation from one task to another one might not be beneficial for a client, because a client might have to interrupt another task instead of the current one. This will lead to the stopping of all donor-task's connected low-layer tasks and insufficient amount of data for its upper-layer tasks. As a result, the utilities of all stopped / interrupted tasks will be decreasing and they may affect the large part of the graph.

The GRA might also penalise a client for this type of re-allocation of resources among tasks by agreeing a shorter period of time for an interrupted task than the remainder of the allocation period of the donor-task. In this way, a re-allocation among the client's own tasks can be beneficial for a client only when it faces a significant loss in its utility (e.g. a system is going to stop). A loss in utility gradually increases towards the end of damping time $\tau_k^{dam}(t_d)$ (see Definition 5.3) for an interrupted task $k$. That is, task's $k$ interruption leads to the larger absolute difference $\Delta P_{k,S_k}(t)$, which shows a growth of error in the client's estimation of parameter $P_{k,S_k}(t)$ (see Definition 5.2). An exceeding of the damping time for task $k$ also means that the delay time $\tau_i^{del}(t_y, t)$ (see Definition 5.4) for each connected (directly or indirectly) recipient-task $i$ can be exceeded, which will lead to the stopping of the larger number of the tasks, connected hierarchically.

Generally, we believe that each interrupted task $k$ should obtain resources before the duration of the current interruption $\breve{\tau}_{k,l}^{int}(t)$ exceeds the damping time $\tau_k^{dam}(t_d)$ or the part $k_{dam} * \tau_k^{dam}(t_d)$ of this time. The part of the damping time may vary depending on how significantly this time affects the client utility. For example, if the utility approaches zero after the damping time is exceeded, than it is reasonable to ensure that a task is run before the end of this time by arranging its swap with a donor-task. In this case, the client would not wait till the end of the damping time to swap one

of its tasks with another one, but it would try to arrange this by negotiating with the GRA before the damping time is passed. A formalised description of the criterion to start negotiation in respect of resource re-allocation for another task is presented below, where $k_{dam}$ is a coefficient which belongs to the interval $[0, 1]$, $t$ is the current time and $l$ is the number of interruption period within the interval of time $[t^0, t_{dl}^{exec}]$.

$$\breve{\tau}_{k,l}^{int}(t) > k_{dam} * \tau_k^{dam}(t_d), \tag{5.17}$$

When the duration of the current interruption period $\breve{\tau}_{k,l}^{int}(t)$ for task, $k$, becomes longer than the specified part $k_{dam}$ of the damping time $\tau_k^{dam}(t_d)$, then a client has to start negotiation with the GRA as for the task's re-allocation with a chosen donor-task.

### 5.3.2 Evaluating the Criteria to Choose the Best Donor-Task

In our work, we assume that every task which belongs to the client can be donated resources from another task which belongs to the same client, if it has a resource. Here, we assume that a resource is a duration of time allocated to execute a specific task. When a client has decided that an interrupted task should be donated a resource from another task, then it has to choose a donor-task which remainder of the allocation period (or a part of this remainder) will be re-assigned to this interrupted task. The issue is to choose which task is the best candidate to be a donor in terms of the less loss in utility, if it loses its allocated time of execution. We distinguish two criteria to choose the best donor-task, where the first one reflects the duration of time which can be allocated for an interrupted task and the second one reflects the dependencies between a donor candidate with other tasks in a graph, which may affect the whole client utility.

#### 5.3.2.1 The Allocation Remainder

The first criterion denotes that an interrupted task prefers to have a longer allocation period, which at least allows its next negotiation to start at the maximum of resource availability. That is, it is desirable for the negotiated duration of time to contain at least one maximum of resource availability. This criterion also considers that the donor-task $j$ as well as all connected tasks will lose significantly in utility, if its remainder $\tau_{j,l}^{rem}(t)$ of the allocation period $\tau_{j,l}^{all}$ at time $t$ is significantly shortened by the GRA during a

negotiation for resource re-allocation between two tasks. Here, a client has to consider "pros" and "cons" of resource re-allocation for both tasks, i.e. the interrupted one and the possible donor-task, because as much this re-allocation might reduce a negative effect from one task's interruption on the client utility as it might increase this effect due to the significant resource loss by another its task.

Therefore, a client is designed to find a donor-task $j$ with a remainder $\tau_{j,l}^{rem}(t)$ of allocation period which is an average $\tau_{av}^{rem}(t)$ between the minimum acceptable $\tau_{min}^{rem}(t)$ and the maximum available $\tau_{max}^{rem}(t)$ remainders among all client tasks at time $t$, which possess resources as described in Formula (5.18).

$$\tau_{av}^{rem}(t) = \left[\tau_{min}^{rem}(t) + \tau_{max}^{rem}(t)\right]/2.0. \tag{5.18}$$

The maximum available remainder $\tau_{max}^{rem}(t)$ is the longest remainder available among all client tasks which possess resources at time $t$. In the ideal scenario, the minimum acceptable duration of task execution should allow a task to start the next negotiation when the resource availability is at its maximum. In this way, the minimum acceptable donor-task's remainder $\tau_{min}^{rem}(t)$ of allocation period should ideally end around the next maximum of resource availability from the current point in time. However, if all available remainders are shorter than this one, we calculate $\tau_{min}^{rem}(t)$ in proportion to the maximum available remainder $\tau_{max}^{rem}(t)$ such as $\tau_{min}^{rem}(t) = k_{rem} \times \tau_{max}^{rem}(t)$, where $k_{rem}$ is a chosen experimentally coefficient from the interval $[0, 1]$. In this case, it is beneficial for the client's task to obtain at least some allocation time instead of prolonging the current interruption.

Figure 5.6 shows an example of the two possible donor-tasks, where task $i$ has an allocation period $\tau_{i,l}^{all}$ and task $j$ has an allocation period $\tau_{j,l}^{all}$. A client has to decide at time $t_{cur}$ whether task $i$ or task $j$ is more preferable for donating their resources to the interrupted task $k$ (both task $i$ and $j$ are running at time $t_{cur}$). If task $i$ donates its resources to the interrupted task $k$, then task $k$ can be run maximum for the duration of time $\tau_{i,l}^{rem}(t_{cur})$, while if task $j$ donates its resources, then task $k$ can be run maximum for the duration of time $\tau_{j,l}^{rem}(t_{cur})$. However, a client aims to start a planned interruption for the re-allocated task $k$ when there is a maximum of resource availability. Therefore, it may shorten the period of time $\tau_{i,l}^{rem}(t_{cur})$ up to the last maximum ("Maximum" in the figure) of resource availability, while the period of time $\tau_{j,l}^{rem}(t_{cur})$ does not allow a client to negotiate within a maximum of resource availability at all (i.e. it does not contain at least one maximum). Therefore, task $j$

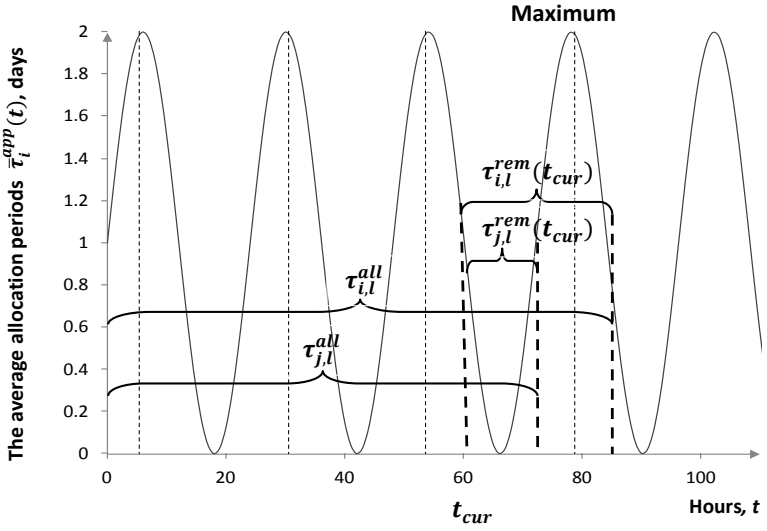



FIGURE 5.6: A comparison of allocation remainders for donor candidates $i$ and $j$

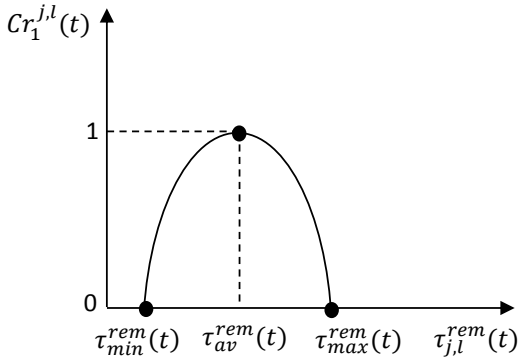



FIGURE 5.7: A function which depicts the first criterion to choose a donor-task

will be less preferable for a client in this case as a donor-task, and task $i$ is more likely to be chosen as a donor for an interrupted task.

According to the first criterion discussed above, a client tends to choose that task as the best candidate for a donor which possesses the remainder of the allocation period, which is closer to the average one $\tau_{av}^{rem}(t)$. Assume $Cr_1^{j,l}(t)$ is a function which formally reflects this preference in the client's decisions and it returns the values from $-\infty$ to one, where one denotes the best donor-task and zero denotes the worst donor-task with the minimum allocation remainder. It has to be noted that in the cases where an allocation remainder is smaller than the minimum acceptable one, this function will return the

negative numbers, which are then algorithmically substituted with zero. However, those tasks are not excluded as possible donor candidates, but they are unlikely to be chosen as a donor. Finally, this function can be described as parabola depicted in Figure 5.7 and presented in Equation (5.19) for the donor candidate $j$ at time $t$.

$$Cr_1^{j,l}(t) = \frac{\left(\tau_{j,l}^{rem}(t) - \tau_{min}^{rem}(t)\right) \times \left(\tau_{max}^{rem}(t) - \tau_{j,l}^{rem}(t)\right)}{\left(\tau_{av}^{rem}(t) - \tau_{min}^{rem}(t)\right) \times \left(\tau_{max}^{rem}(t) - \tau_{av}^{rem}(t)\right)}. \tag{5.19}$$

#### 5.3.2.2 The Donor-Task's Dependencies

The second criterion denotes that resource re-allocation from a donor task should the least affect the utility of the client system, compared to the other donor candidates. Assume that a client has the list of donor candidates and each of them has some remaining time till its planned interruption. However, the data from these candidates has the different levels of importance in respect of their corresponding recipient-task(s). For example, one of the donor candidates monitors the level of temperature in a small usually unattended room in the building, while another donor candidate monitors the level of temperature in a large conference room. Both of them send this data to a particular recipient-task, which monitors then the temperature level on the whole floor. In this case, an interruption of the second donor candidate will result in a more significant loss in the client utility than an interruption of the first donor candidate in terms of their impact on the recipient-task results, if both of those candidates are running tasks. Intuitively, the first candidate is more suitable for a client to use as a donor than the second candidate.

Assume that task $j$ is a donor candidate and it has a recipient-task $i$ for which the level of importance of its data is measured by $\alpha_{j,i}$ (see Section 5.2.1). A delay time $\tau_i^{del}(t_y, t)$ (see Equation (5.11)) for the corresponding recipient-task $i$ in the case task $j$ is interrupted is longer, if $\alpha_{j,i}$ is smaller (i.e. the sender-task is less important for this recipient-task). In this way, if the recipient-task of the donor candidate is running, then a client has to estimate when this task will be stopped due to an inaccurate input data, considering the corresponding donor candidate is interrupted (if not already interrupted). Consequently, a client prefers that task as a donor which has the longest remaining time of execution of its corresponding recipient-task in case this donor candidate is accepted as a donor-task. This remaining period of time can be equal to the delay time $\tau_i^{del}(t_y, t)$, if the recipient-task $i$ is currently running with

an accurate input data, or it can be equal to the remaining time of the delay time $\tau_i^{rdel}(t_y, t) = \tau_i^{del}(t_y, t) - (t_y - t)$ at time $t$, if the recipient-task $i$ is currently running with an inaccurate input data and its sender-task $j$, which is considered for a donor, can be running or not at the moment. The longer $\tau_i^{rdel}(t_y, t)$ for the recipient-task $i$ of the donor candidate $j$, the more preferable this donor candidate for a client.

We also consider the situation when the donor candidate's recipient-task is not running (i.e. it might be stopped or interrupted) at the time of choosing a donor-task. In this case, we cannot estimate the delay time of this recipient-task, because it has already been stopped or interrupted. However, this recipient-task might potentially be run again and, therefore, we have to estimate how the interruption of its sender-task (which is a donor candidate) might affect its execution. In this situation, a client considers the level of importance $\alpha_{j,i}$ of a donor candidate $j$ in respect of its recipient-task $i$. The closer this level of importance $\alpha_{j,i}$ to the minimal importance level $\alpha_{min}$, compared to the maximal importance level $\alpha_{max}$ among all client tasks (except for the root task), the more preferable task $j$ as a donor candidate. Considering the client preferences described above, we can determine a variable $Con_j^i(t_y, t)$ at time $t$ for each donor candidate $j$, which value varies between zero and one, where one denotes the most preferable donor candidate and zero denotes the least preferable donor candidate. Formally, those preferences at time $t$ are presented in Formula (5.20), where task $i$ represents a recipient-task of a donor candidate $j$ (i.e. $j \in S_i$).

$$Con_j^i(t_y, t) = \begin{cases} \dfrac{\tau_i^{rdel}(t_y, t) - \tau_{min}^{rdel}(t)}{\tau_{max}^{rdel}(t) - \tau_{min}^{rdel}(t)}, & \textit{when task } i \textit{ is running}, \\ \dfrac{\alpha_{max} - \alpha_{j,i}}{\alpha_{max} - \alpha_{min}}, & \textit{when task } i \textit{ is not running}. \end{cases} \tag{5.20}$$

We also assume that the donor candidates from the lower layers of a tree are more preferable for a client compared to the donor candidates from the upper layers, because interruption of an upper layer task will decrease the client utility more significantly than interruption of a lower layer task. The higher the layer of the task, the larger number of the lower layer tasks will be stopped if this higher layer task is interrupted. Moreover, the higher layer tasks are considered as more important for a client than the lower layer tasks, because they are expected to perform more sophisticated operations and to control the larger part of the client system. For instance, the higher layer task might be responsible for the control of the temperature level on the whole floor of the building, while the lower layer task might be responsible for the control of the temperature level

in a single room on this floor. The root task indicates the highest layer $N_{lay} - 1$, while the lowest layer of a tree is identified as a zero layer. Then, we define a variable $Lay_j$, which value varies between zero and one, where one determines the most preferable donor candidate and zero determines the least preferable donor candidate. Finally, $Lay_j$ is defined as below:

$$Lay_j = 1 - \frac{Layer_j}{N_{lay} - 1}, \tag{5.21}$$

where $Layer_j = 0$ denotes the most preferable layer, and $Layer_j = N_{lay} - 1$ denotes the least preferable one.

The last criterion of the client to choose a donor-task is a status of execution $Status_j(t)$ (see Definition 5.1) of a donor candidate at the current moment of time, which denotes whether a task is running (with or without accurate input data) or not running. The status of the task depends on the other tasks in a tree e.g., whether other connected tasks are running or not running. A client do not consider the tasks with a status 'interrupted' as the possible donor candidates, because they do not possess any resources, but it considers all other tasks with the statuses 'stopped', 'inaccurate' or 'accurate' (see Section 5.2.1). A status 'stopped' of a donor candidate is considered as the most preferable for a client in terms of the least negative impact on the client utility. That is, if the task with such execution status is chosen as a donor, it does not need to be interrupted because it is already stopped. However, if a donor candidate has a status 'inaccurate' or 'accurate' and it will be stopped, then this will affect negatively all other connected tasks which were running without or smaller error before. That is, the statuses 'inaccurate' and 'accurate' are regarded as equally non-preferable statuses. Finally, we introduce a variable $Stat_j(t)$ for a donor candidate $j$ in the following Formula (5.22):

$$Stat_j(t) = \begin{cases} 0, \; if \;\; Status_j(t) = `interrupted', \\ 0.6, \; if \;\; Status_j(t) = `inaccurate' \vee `accurate', \\ 1, \; if \;\; Status_j(t) = `stopped'. \end{cases} \tag{5.22}$$

Consequently, a function $Cr_2^{j,i}(t_y, t)$, $j \in S_i$, $j \neq i$ determines the overall client decision in terms of the values from zero to one, considering client preferences mentioned in this section and it is described in the following equation:

$$Cr_2^{j,i}(t_y, t) = Con_j^i(t_y, t) \times Lay_j \times Stat_j(t). \tag{5.23}$$

#### 5.3.2.3 The Overall Decision Making Function

A function $Cr_{all}^{j,i,l}(t_y, t)$, which produces a value from zero to one for each donor candidate $j$, combines all client criteria based on which a client is expected to make a balanced decision. The two criteria $Cr_1^{j,l}(t)$ and $Cr_2^{j,i}(t_y, t)$, described in Equations (5.19) and (5.23) respectively, have the weights $W_1$ and $W_2$ which determine their level of impact on the client's decision, and $Cr_{all}^{j,i,l}(t_y, t)$ is presented in the following equation:

$$Cr_{all}^{j,i,l}(t_y, t) = W_1 \times Cr_1^{j,l}(t) + W_2 \times Cr_2^{j,i}(t_y, t), \tag{5.24}$$

where $W_1$ and $W_2 \in [0, 1]$ and their sum is equal to one.

### 5.3.3 Algorithm for Task Re-allocation

As we discussed above, a client has to decide when it is necessary to donate a resource to an interrupted task from a chosen donor task and how to choose the best donor task. A client also has to negotiate with the GRA in respect of its decision to re-allocate a resource from one task to another one, because only the GRA has an authority and knowledge about physical Grid resources to re-allocate a particular task. If resources are significantly demanded in the Grid, then the GRA might become greedy in negotiation and this may lead to a failure of negotiation or the shorter allocation time than the allocation remainder of the donor-task. Hence, the client's request to swap resources between its tasks might be partially satisfied by the GRA or not satisfied at all.

In our work, we distinguish between the expected and unexpected interruptions, where the unexpected interruptions may occur randomly with some probability $Prob$, and the expected ones occur when an allocation period $\tau_{i,l}^{all}$ has passed as described in Algorithm 5.3. For example, if a randomly generated number *Random number* smaller than $Prob = 0.001$, then the status of a task $i_1$ has to be changed to 'interrupted'. Then all other tasks $i_2$ have to change their statuses accordingly. That is, the lower layer tasks (if applicable) which are connected to task $i_1$ directly or indirectly have to change their statuses to 'stopped' unless they have had a status 'interrupted', while the upper layer tasks (if applicable) have to change their statuses to 'inaccurate' if they had a status 'accurate' before task $i_1$ has been interrupted. The change of statuses also incorporates a calculation of damping and delay times according to those statuses as described in Sections 5.2.2 and 5.2.3.

**Algorithm 5.3** The change in tasks' statuses when task $i_1$ is interrupted

1: **for** Each task $i_1 \in Tr$ **do**
2: **if** ($Random\ number < Prob$) OR $\left(\tau_{i_1,l}^{all}\ is\ finished\right)$ **then**
3: Change $Status_{i_1}(t)$ to 'interrupted'
4: **for** Each task $i_2 \in Tr,\ i_2 \neq i_1$ **do**
5: **if** $(Status_{i_2}(t) \neq$ '$interrupted$') AND $(Layer_{i_2} < Layer_{i_1})$ AND ($i_2$ *is connected with* $i_1$) **then**
6: Change $Status_{i_2}(t)$ to 'stopped' (see Section 5.2.1)
7: **end if**
8: **if** $(Status_{i_2}(t) =$ '$accurate$') AND $(Layer_{i_2} > Layer_{i_1})$ AND ($i_2$ *is connected with* $i_1$) **then**
9: Change $Status_{i_2}(t)$ to 'inaccurate' (see Section 5.2.1)
10: **end if**
11: **end for**
12: **end if**
13: **end for**

**Algorithm 5.4** The change in tasks' statuses when task $i_1$ has obtained a resource

1: {Task $i_1$ has just obtained a resource, i.e. $Status_{i_1}(t) =$ '$accurate$'}
2: **for** Each task $i_2 \in Tr,\ i_2 \neq i_1$ **do**
3: **if** $Status_{i_2}(t) \neq$ '$interrupted$' **then**
4: Change $Status_{i_2}(t)$ to 'accurate' {This status is assigned temporarily}
5: **end if**
6: **end for**
7: **for** Each task $i_3 \in Tr$ **do**
8: **if** $Status_{i_3}(t) =$ '$interrupted$' **then**
9: Repeat lines 4-11 of Algorithm 5.3, if $i_2 \neq i_3$
10: {Here, task $i_1$ might change its status to 'inaccurate' due to the interrupted sender-tasks.}
11: **end if**
12: **end for**

We also model a change of other tasks' statuses in the case when task $i_1$ obtains resources through negotiation with the GRA, which is described in Algorithm 5.4. Initially, the status of this task changes to 'accurate' nominally. However, this status is assigned temporarily until a client discovers whether this task can be run with an accurate input data or it has some interrupted direct or indirect sender-tasks, which will cause the change of its status to 'inaccurate'. In addition, if task's $i_1$ interrupted sender-tasks cause the large error in its estimation of parameter $P_{i_1,S_{i_1}}(t)$ (i.e. $\Delta P_{i_1,S_{i_1}}(t) > \eta_{i_1}^{del}$ ), this will lead to the change of its status to 'stopped' from the 'accurate' or 'inaccurate' statuses, which is presented in Algorithm 5.5. In this algorithm, $i_3$ can

**Algorithm 5.5** The change in tasks' statuses due to the large difference $\Delta P_{i_1,S_{i_1}}(t)$

1: **repeat**
2: Label = 0
3: **for** Each task $i_3 \in Tr$ **do**
4: **if** $\Delta P_{i_3,S_{i_3}}(t) > \eta_{i_3}^{del}$ **then**
5: Change $Status_{i_3}(t)$ to 'stopped'
6: Repeat lines 4-11 of Algorithm 5.3, if $i_2 \neq i_3$ {A status 'stopped' is considered as 'interrupted' in terms of the increase in $\Delta P_{i_3,S_{i_3}}(t)$}
7: Label = 1
8: **end if**
9: **end for**
10: **until** Label = 0

be equal to $i_1$ as well, i.e. this loop checks the absolute difference $\Delta P_{i_3,S_{i_3}}(t)$ for all client tasks, considering which tasks have to be stopped due to the large error in their parameter's estimation, and then it changes all other tasks' statuses according to this change. This algorithm also re-checks $\Delta P_{i_3,S_{i_3}}(t)$ for each task in case the stopped tasks have caused a significant increase in $\Delta P_{i_3,S_{i_3}}(t)$ for other tasks until no running tasks are found to exceed $\eta_{i_3}^{del}$, which is regulated by *Label*.

Finally, the whole algorithm of resource re-allocation from one client's task to another one is presented in Algorithm 5.6. Here, the statuses of all tasks are re-assigned if necessary, according to their execution status, every virtual time unit $t$ as presented in lines 4-9, where a Boolean variable OBTAINED becomes true if at least one task has obtained an allocation period during previous time unit. Then our algorithm calculates the values of $Cr_{all}^{i_3,i,l}(t_y,t)$ for each task $i_3$ which is not interrupted at time $t$, and the maximal value $Cr_{all}^{max}(t_y,t)$ is chosen, indicating the best donor candidate $i_{best}$. Consequently, a client starts negotiating with the GRA in respect of the allocation period $\hat{\tau}_{i_3,r_{i_3}}^{all}(t)$ at time $t$ in round $r_{i_3}$, which is counted separately for each task $i_3$. For example, the largest number of negotiation rounds for a single negotiation can be set to 100 rounds and every time it fails, it starts again from round zero.

In our evaluation, the proposals of a client and the GRA are assumed to be generated according to the time-dependent concession-based negotiation strategies which are described in Chapters 3 and 4. However, we do not focus on the negotiation strategies in this chapter, and we assume that any strategy can be used here. If after one exchange of proposals in turns neither side has accepted an opponent's proposal and a criterion presented in Formula (5.17) is satisfied for task $i_3$, then a resource re-allocation

**Algorithm 5.6** Client's SimTask re-allocation strategy based on $Cr_{all}^{i_3,i,l}(t_y,t)$

1: OBTAINED = FALSE
2: **repeat**
3: Each virtual time unit $t$
4: **if** OBTAINED==TRUE **then**
5: Algorithm 5.4
6: OBTAINED = FALSE
7: **end if**
8: Algorithm 5.3
9: Algorithm 5.5
10: $Cr_{all}^{max}(t_y,t)=0$
11: **for** Each task $i_3 \in Tr$ with $Status_{i_3}(t) \neq$ '$interrupted$' **do**
12: Calculate $Cr_{all}^{i_3,i,l}(t_y,t)$ (see Section 5.3.2)
13: **if** $Cr_{all}^{i_3,i,l}(t_y,t) > Cr_{all}^{max}(t_y,t)$ **then**
14: $Cr_{all}^{max}(t_y,t) = Cr_{all}^{i_3,i,l}(t_y,t)$
15: $i_{best} = i_3$
16: **end if**
17: **end for**
18: **for** Each task $i_3 \in Tr$ with $Status_{i_3}(t) =$ '$interrupted$' **do**
19: Exchange the proposals with the GRA in respect of $\hat{\tau}_{i_3,r_{i_3}}^{all}(t)$
20: **if** Agreement is reached **then**
21: Change $Status_{i_3}(t)$ to 'accurate' {This status is assigned temporarily}
22: OBTAINED = TRUE
23: **end if**
24: **if** $\left(\breve{\tau}_{i_3,l}^{int}(t) > k_{dam} * \tau_{i_3}^{dam}(t_d)\right)$ AND $(Status_{i_3}(t) =$ '$interrupted$') AND $(Status_{i_{best}}(t) \neq$ '$interrupted$') **then**
25: Exchange the proposals with the GRA in respect of $\tau_{i_{best},l}^{rem}(t)$
26: **if** Agreement is reached **then**
27: Change $Status_{i_3}(t)$ to 'accurate' {This status is assigned temporarily}
28: OBTAINED = TRUE
29: Change $Status_{i_{best}}(t)$ to 'interrupted'
30: **end if**
31: **end if**
32: **end for**
33: **until** $t_{dl}^{exec}$ is reached

among client tasks can be considered as described in lines 24-31. However, the resource re-allocation among tasks also have to be agreed with the GRA, and this negotiation might also fail. In our algorithm, a remainder of allocation period $\tau_{i_{best},l}^{rem}(t)$ of a chosen donor-task $i_{best}$ is negotiated for each interrupted task $i_3$ at time $t$, and the first one which reaches an agreement with the GRA will obtain the agreed part of this remainder

(may be the whole remainder). At the same time, all tasks which do not possess resources continue to negotiate with the GRA about available Grid resources every next time unit. In this way, each interrupted task may obtain an allocation period from the GRA through ordinary negotiation or negotiation for allocation remainders to swap its own tasks every time unit $t$.

**Example for Algorithm 5.6.**

This example shows how a donor-task is chosen by a client using the SimTask re-allocation strategy. Assume that we have 40 tasks, which are connected as presented in Figure 5.8, where the root task has index 0 and the lowest layer tasks belong to the layer zero. The lower layer tasks send data to the upper layer tasks as we discussed above. Then, assume that task 31 changes its status of execution to 'interrupted', then tasks 11, 3 and 0 change their statuses to 'inaccurate', while all other tasks have a status 'accurate'. Any task which possess resources can potentially be chosen as a donor-task for task 31.

First, a re-allocation algorithm assesses the remaining allocation time for all tasks, where the maximum available allocation remainder is 338290.0 (virtual seconds) and the minimum acceptable allocation remainder is 129600.0 (virtual seconds). Here, the preference is given to the allocation remainders, which are closer to the average allocation remainder between the maximum available and the minimum acceptable remainders. Then, the allocation remainders for some tasks are presented below.

| Task | The allocation remainder (virtual seconds) |
|---|---|
| 0 | 204942.0 |
| 1 | 312833.0 |
| 2 | 214528.0 |
| ... | ... |
| 6 | 220063.0 |
| 7 | 73042.7 |
| ... | ... |
| 32 | 338290.0 |
| 33 | 139458.0 |
| 34 | 242676.0 |
| ... | ... |
| 39 | 165681.0 |

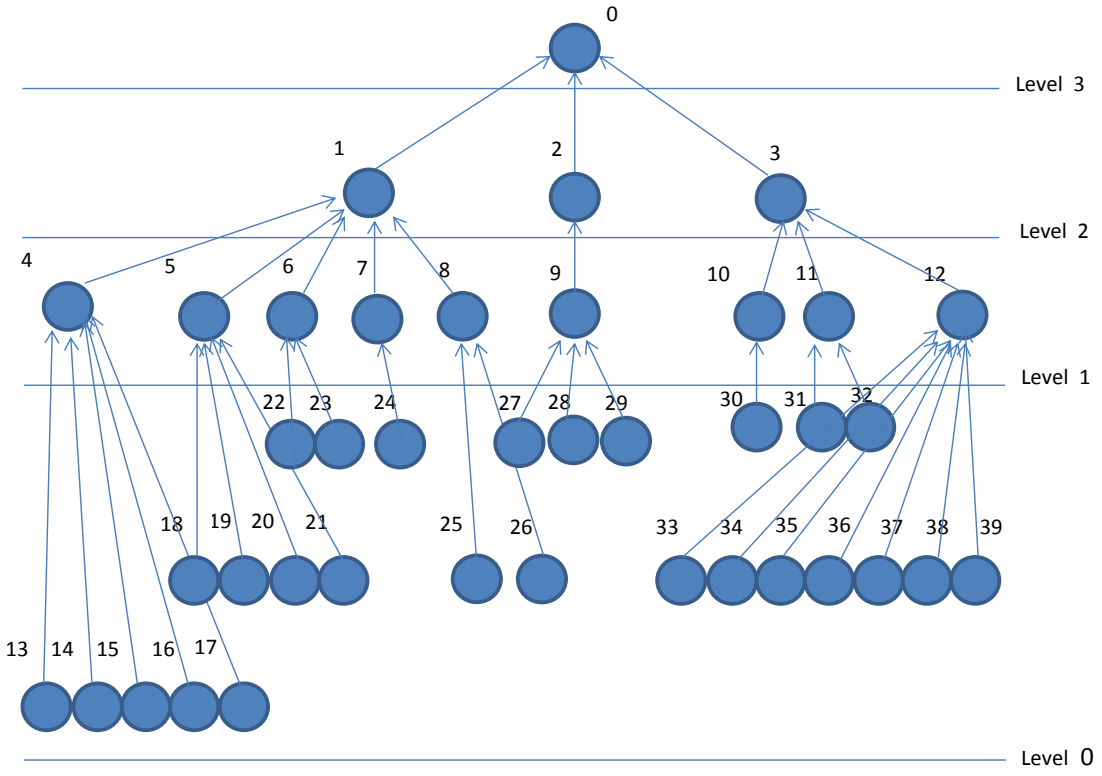



FIGURE 5.8: An example of a tree of tasks

Second, a re-allocation algorithm assesses which donor candidate's interruption will have the least negative effect on the tree of tasks. Here, the preference is given to the lowest layer tasks, the tasks with the status 'stopped' and the tasks with the longer possible delay times (or remaining delay times) for their corresponding recipient-tasks in the case they are chosen to be a donor. In our example, the shortest delay time will be for task 2 if task 9 is chosen as a donor and it equals to 67.3064 (virtual seconds). The longest delay time will be for task 12 if task 34 is chosen as a donor and it equals to 730720.0 (virtual seconds). The task 34 also belongs to the lowest layer of the tree, and its allocation remainder is close to the average remainder as listed above.

In this example, the client prioritises a donor candidate's impact on the tree of tasks, if it is chosen as a donor, over its allocation remainder, i.e. $W_1 = 0.3$ and $W_2 = 0.7$ in Formula (5.24). Therefore, the task 34 is chosen as a donor.

TABLE 5.2: List of notation for Section 5.3

| Symbol | Notation |
|---|---|
| $\breve{\tau}_{i,l}^{int}(t)$ | A duration of interruption with the identifier $l$ for task $i$ at time $t$, where $i, l \in \mathbb{N}$, $t \in \mathbb{R}$. |
| $k_{dam}$ | A coefficient which indicates the level of tolerance of a client towards approaching end of a damping time, where $k_{dam} \in [0, 1]$. |
| $\tau_{j,l}^{rem}(t)$ | A remainder of an allocation period $\tau_{j,l}^{all}$ at time $t$ for task $j$, where $\tau_{j,l}^{rem}(t) > 0$, $j, l \in \mathbb{N}$, $t \in \mathbb{R}$. |
| $\tau_{min}^{rem}(t)$ | A minimal desirable remainder of an allocation period at time $t$, where $\tau_{min}^{rem}(t) > 0$, $t \in \mathbb{R}$. |
| $\tau_{max}^{rem}(t)$ | A maximal remainder of an allocation period at time $t$ among all tasks which possess resources, where $\tau_{max}^{rem}(t) > 0$, $t \in \mathbb{R}$. |
| $\tau_{av}^{rem}(t)$ | An average remainder between minimal $\tau_{min}^{rem}(t)$ and maximal $\tau_{max}^{rem}(t)$ remainders, where $\tau_{av}^{rem}(t) > 0$, $t \in \mathbb{R}$. |
| $Cr_1^{j,l}(t)$ | A criterion which produces values from zero to one, indicating a client's preference in respect of the remainder of allocation period $\tau_{j,l}^{rem}(t)$ of a potential donor-task $j$, where $Cr_1^{j,l}(t) \in [0, 1]$, $t \in \mathbb{R}$, $j, l \in \mathbb{N}$. Here, one denotes the most desirable remainder, while zero denotes the least desirable remainder. |
| $\tau_i^{rdel}(t_y, t)$ | A remainder of delay time $\tau_i^{del}(t_y, t)$ for a recipient-task $i$ at time $t$, which sender-task is a donor candidate, where $\tau_i^{rdel}(t_y, t) > 0$, $t, t_y \in \mathbb{R}$, $i \in \mathbb{N}$. |
| $\tau_{min}^{rdel}(t)$ | A minimal remainder of delay time among all tasks which corresponding recipient-tasks are not stopped or interrupted, where $\tau_{min}^{rdel}(t) > 0$, $t \in \mathbb{R}$. |
| $\tau_{max}^{rdel}(t)$ | A maximal remainder of delay time among all tasks which corresponding recipient-tasks are not stopped or interrupted, where $\tau_{max}^{rdel}(t) > 0$, $t \in \mathbb{R}$. |
| $\alpha_{max}$ | A maximal level of importance $\alpha_{j,i}$ among all pairs of a sender-task $j$ and recipient-task $i$, where $\alpha_{max} \in [0, 1]$, $i, j \in \mathbb{N}$. |
| $\alpha_{min}$ | A minimal level of importance $\alpha_{j,i}$ among all pairs of a sender-task $j$ and recipient-task $i$, where $\alpha_{min} \in [0, 1]$, $i, j \in \mathbb{N}$. |

Continued on the next page

Continued from the previous page

| **Symbol** | **Notation** |
|---|---|
| $Con_j^i\,(t_y, t)$ | A variable which produces values from zero to one, determining a client's judgment of a donor candidate's $j$ impact on the corresponding recipient-task $i$ in the case if its resource is re-allocated to another task, measured in $\tau_i^{rdel}\,(t_y, t)$ or $\alpha_{j,i}$, where $i, j \in \mathbb{N}$, $t, t_y \in \mathbb{R}$. |
| $Lay_j$ | A variable which produces values from zero to one, determining a client's preference in respect of a layer $Layer_j$ to which a donor candidate $j$ belongs to, where $j \in \mathbb{N}$. |
| $Stat_j\,(t)$ | A variable which produces values from zero to one, determining a client's preference in respect of the status of execution of a donor candidate $j$, where $t \in \mathbb{R}$, $j \in \mathbb{N}$. |
| $Cr_2^{j,i}\,(t_y, t)$ | A criterion which produces values from zero to one, indicating a client's preference in respect of a donor candidate's $j$ impact on the corresponding recipient-task $i$, its layer and status of execution, where $i, j \in \mathbb{N}$, $t, t_y \in \mathbb{R}$. |
| $W_1,\ W_2$ | The weight coefficients which determine the level of impact of each criterion $Cr_1^{j,l}\,(t)$ or $Cr_2^{j,i}\,(t_y, t)$ on the client's decision, $W_1,\ W_2 \in [0, 1]$, $W_1 + W_2 = 1.0$. |
| $Cr_{all}^{j,i,l}\,(t_y, t)$ | A function which determines the best donor candidate $j$ by combining the criteria $Cr_1^{j,l}\,(t)$ and $Cr_2^{j,i}\,(t_y, t)$, where $Cr_{all}^{j,i,l}\,(t_y, t) \in [0, 1]$, $t, t_y \in \mathbb{R}$, $i, j, l \in \mathbb{N}$. |

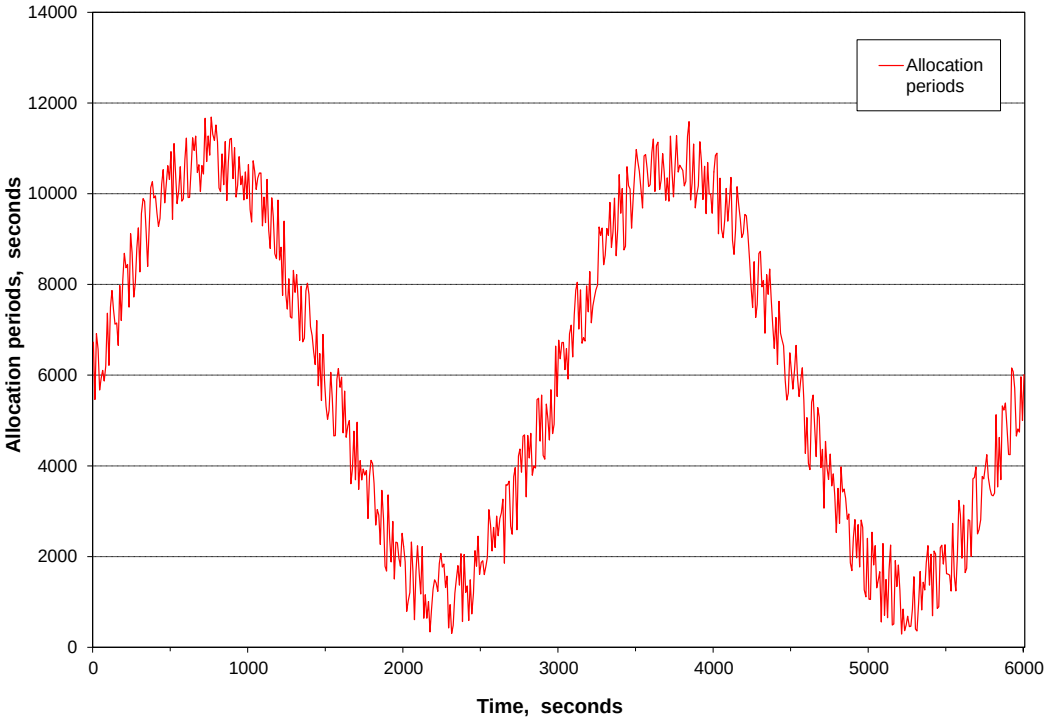
## 5.4 Results and Discussion

In this section, we evaluate the SimTask re-allocation strategy in terms of the client utility, compared to the case when this strategy is not used, in various Grid environments with the different priorities in respect of criteria to choose a donor-task. The different environments are modelled by varying the probability of unexpected task interruption and an accuracy of the client's estimation of the resource availability maximum as this accuracy shows the level of periodical determinism in the resource availability fluctuations. The probability of task unexpected interruption denotes the level of reliability and stability of the Grid system in terms of the possibility of resource failure and / or withdrawal. That is, the more reliable Grid system assumes the less number of

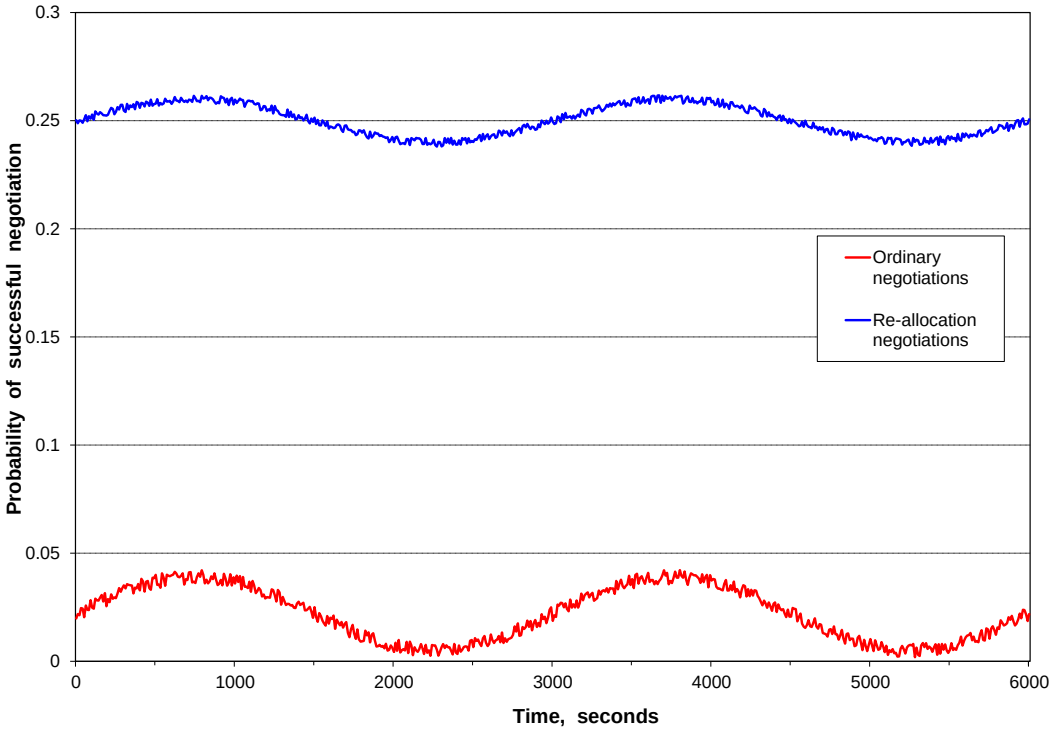
resource failures, and the more stable Grid system assumes the less number of resource withdrawals per time unit. As a result, the more favourable Grid environment for task execution has the smaller probability of task unexpected interruption. We also consider that a client is able to identify the maximum of resource availability to some degree, which depends on the different conditions such as the level of determinism of periodicity in resource availability, the amplitude of random oscillations in the resource availability, etc. The more accurate a client is able to identify the maximum of resource availability, the more favourable conditions are for negotiation during the expected interruptions of the tasks. The different priorities over criteria to choose the best donor-task denote that the most suitable remaining allocation period $\tau_{j,l}^{rem}(t)$ or the least influential donor candidate in respect of other tasks' execution affects the client decision to some degree which might be different for each of them. In other words, we evaluate the client's utility in respect of the different criteria weights $W_1$ and $W_2$ to choose the best donor-task as described in Equation (5.24).

In our evaluation, a client possesses 40 tasks which are connected hierarchically, where each task has three sender-tasks (if applicable) respectively. We have chosen a four-layer tree, because it has an internal layer which is not directly connected to the lowest layer. That is, this tree has the root, the lowest and two intermediate layers. A tree has been modelled as symmetric in order to make its testing more intuitive. The values of $\alpha_{i,j}$ are generated randomly for each test, which means that all tasks have to be run continuously and simultaneously for $\tau_{dl}^{exec} = 300000$ virtual seconds. The period of resource availability changes is equal to 3000 virtual seconds. If we translate these virtual seconds into the virtual minutes or days, then 3000 virtual seconds are assumed to be 50 virtual minutes, while 300000 virtual seconds constitute more than 3 virtual days. Thee virtual days have been considered as a long task execution, compared to the period of resource availability fluctuation. The average client utility is than calculated over 200 runs.

A possibility for a task to obtain the longer duration of an allocation period is simulated, following a periodicity of resource availability as depicted in Figure 5.9(a), where these durations fluctuate periodically over time. The probability of successful negotiation also increases when resources are more available as presented in Figure 5.9(b). We also assume that in the resource re-allocation negotiation the GRA is less greedy than in the ordinary negotiation, because it does not need to allocate free resources, which can be also required by other clients, but only re-allocate the resources which have already been granted to a client for another client's task. Therefore, the probability of successful

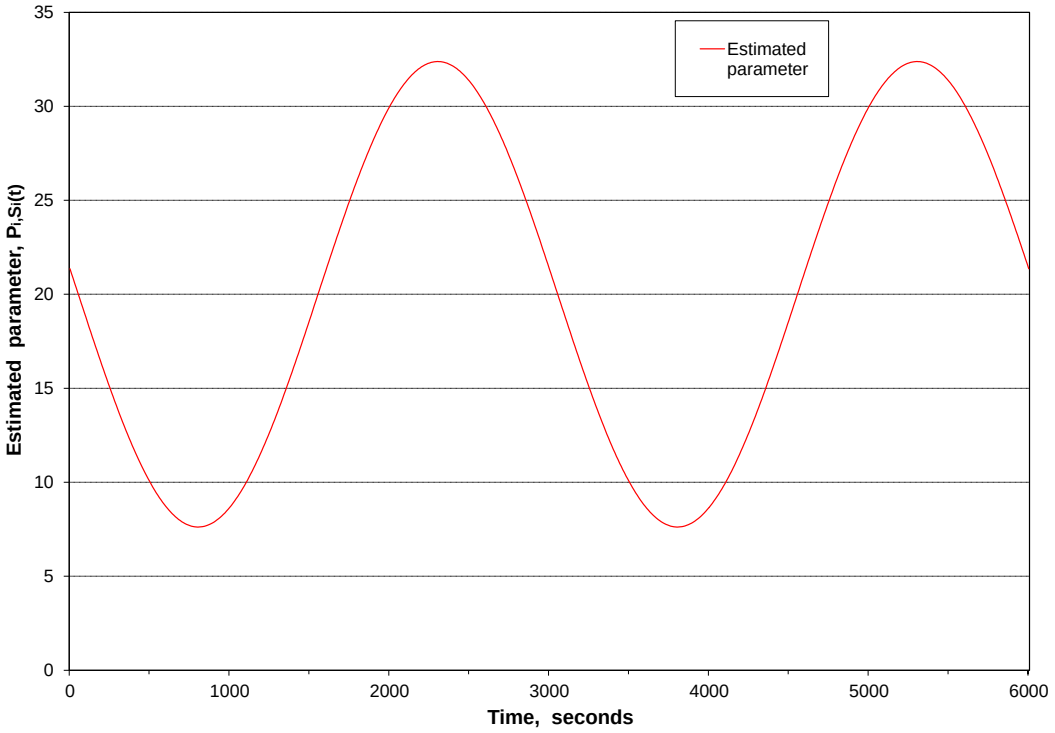



(a) A periodicity of the possible allocation periods over time

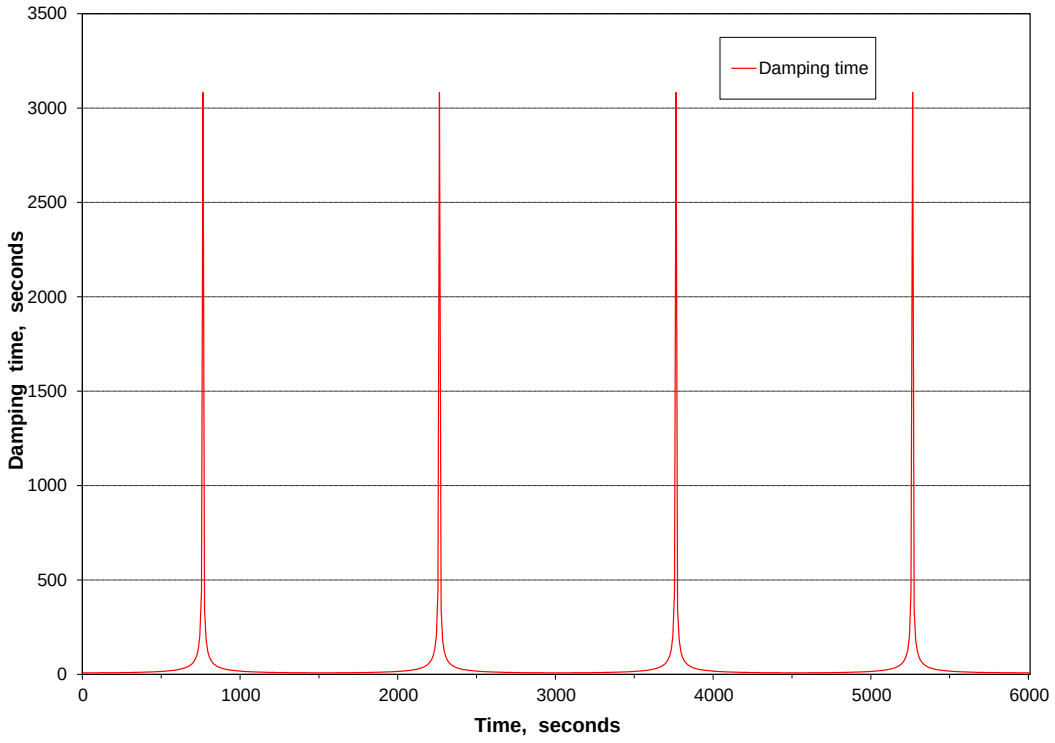



(b) A probability of successful negotiation over time

FIGURE 5.9: A simulation of negotiation outcomes




(a) The change of the estimated parameter over time (e.g. temperature in Celsius)




(b) The change of the damping time over time

FIGURE 5.10: A simulation of monitored (controlled) environment

negotiation for the inter-task resource re-allocation is modelled to be generally higher than this probability for the ordinary negotiation with the GRA as depicted in Figure 5.9(b). Here, the average duration of interruption constitutes around 10 or 20 virtual seconds, but it can be much longer when resources are scarce.

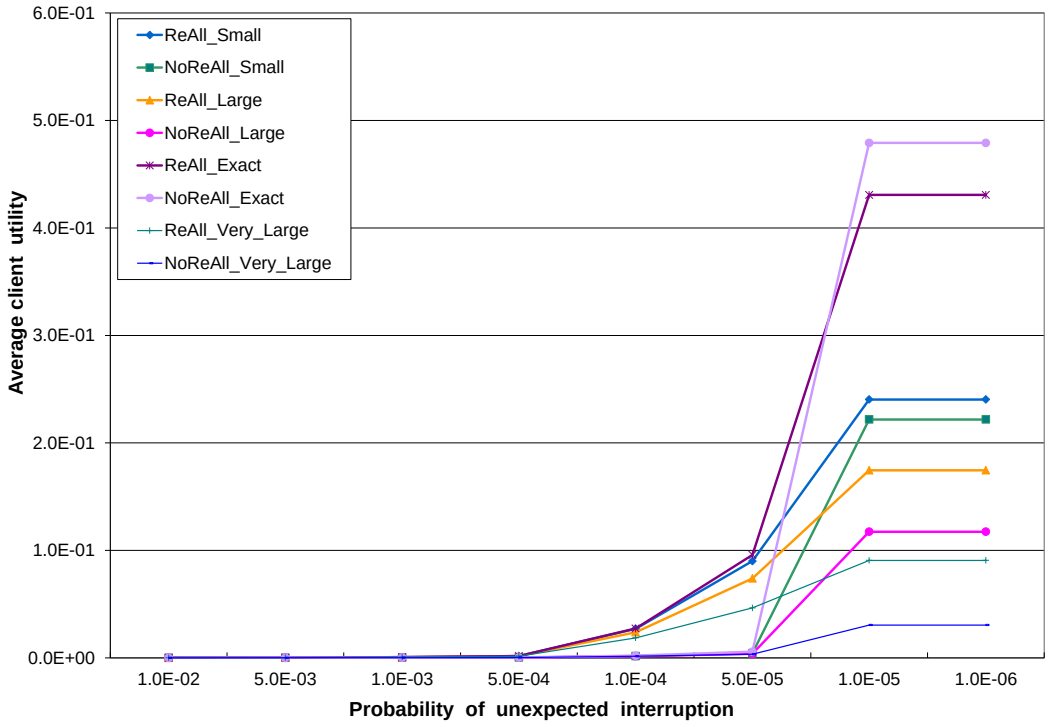
The change of an estimated parameter $P_{i,S_i}(t)$ (see Definition 5.2) over time is modelled as a periodic function, because we assume that this parameter may change, depending on the time of the day, season, etc. (e.g. the temperature level)[1]. For instance, an outside temperature at night is always lower than during a day-time. This function is presented as in Figure 5.10(a). It has to be noted that the damping time, which is estimated in Equation (5.10) and presented in Figure 5.10(b), has peaks at the minimum and maximum of the function $P_{i,S_i}(t)$, because the derivatives of $P_{i,S_i}(t)$ are close to zero at these peaks. This means that the parameter $P_{i,S_i}(t)$ changes slowly at its peaks and, therefore, the error in the client's estimation of this parameter rises slowly over time which leads to the longer damping times.

### 5.4.1 Re-allocating Tasks in the Different Grid Environments

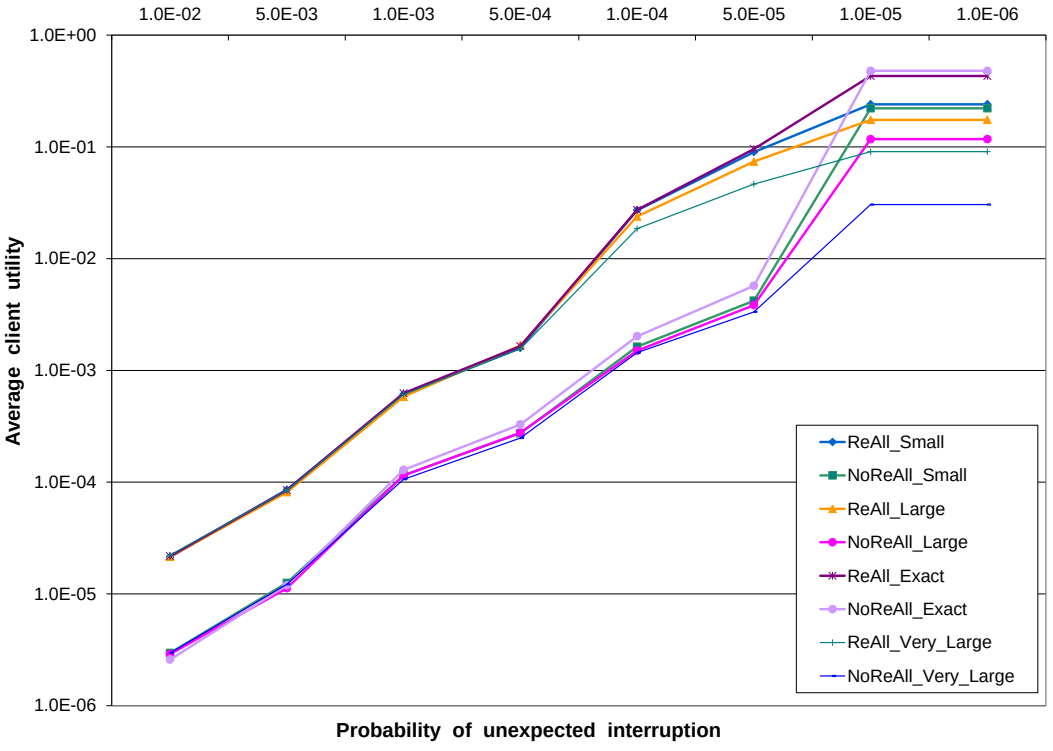
In this section, we evaluate the change in the client's utility for the different Grid environments, i.e. the different probabilities of task unexpected interruption and the different levels of accuracy with which a client is able to estimate the maximum of resource availability. The probabilities of task unexpected interruption are considered in the interval from 1.E-02 till 1.E-06. The probabilities larger than 1.E-02 are considered to be non-realistic, because all tasks would be interrupted so often that the client system would not run adequately (such as almost every virtual second). Figure 5.11(a) supports this assumption as it shows that the client utility changes insignificantly above the probability 5.E-04 and it generally tends to zero towards the larger probabilities. We also do not model the probabilities below 1.E-06, because Figures 5.11(a) and 5.11(b) show that the client utility does not noticeably change below probability 1.E-05. This occurs due to the fewer number of task unexpected interruptions which is approximately the same for such small probabilities and any possible difference averages over multiple runs.

The different level of accuracy with which a client is able to estimate the maximum of resource availability mean that it will be more difficult for a client to obtain allocation

[1]The change of this parameter can be modelled with another functional dependence, if it satisfies Definition 5.2.




(a) The average client utility over the probability of task unexpected interruption




(b) The average client utility in a logarithmic scale over the probability of task unexpected interruption

FIGURE 5.11: The changes in the client utility in the different Grid environments

period (as a resource) through an ordinary negotiation with the GRA, as a client might not be able to start its planned interruptions at the maximum of resource availability. In this case, a client is more likely to use the SimTask re-allocation strategy in order to run the interrupted tasks. We consider four different levels of accuracy in the estimation of the maximum resource availability by a client, where a precise estimation is indicated as "ReAll_Exact" and "NoReAll_Exact", while an inaccurate estimation with a small deviation is indicated as "ReAll_Small" and "NoReAll_Small", with a large deviation as "ReAll_Large" and "NoReAll_Large", and with a very large deviation as "ReAll_Very_Large" and "NoReAll_Very_Large" in Figure 5.11. Here, a small deviation (positive or negative) from a maximum of resource availability is considered to be up to 1% of the duration of a period of resource availability fluctuation. A large and very large deviations denote up to 2% and 4% of the duration of one virtual day respectively. The choice of those values aims to evaluate a trend in the client utility over the client's possibility to estimate maximum resource availability. Finally, we compare the cases when a client uses a SimTask re-allocation strategy (i.e. "ReAll") and when it does not use this strategy (i.e. "NoReAll"). In all cases "ReAll", the client weights in respect of the two criteria to determine the best donor-task are chosen to be $W_1 = 0.3$ and $W_2 = 0.7$ (see Equation (5.24)) as the most successful combination among all considered combinations. The different weights are tested in the following section.

Figure 5.11 shows the average client utilities for the different probabilities of task unexpected interruption in a conventional (see Figure 5.11(a)) and logarithmic (see Figure 5.11(b)) scale. It has to be noted that the SimTask re-allocation strategy improves the client utility in almost all modelled cases, except for the two smallest probabilities when the maximum of resource availability can be estimated precisely. This just shows that in such cases the necessity to use this re-allocation strategy decreases, when the number of unexpected interruptions drastically drops. Hence, this re-allocation strategy might decrease the client's utility due to unnecessary interruptions of the donor-tasks. However, in the cases of the small, large or very large deviations in the client's estimation of the resource availability maximum, the SimTask re-allocation strategy shows a noticeable improvement (especially, for the larger deviations) in the client's utility over the cases when this strategy is not used even for the small probabilities of unexpected interruption such as 1.E-05 and 1.E-06.

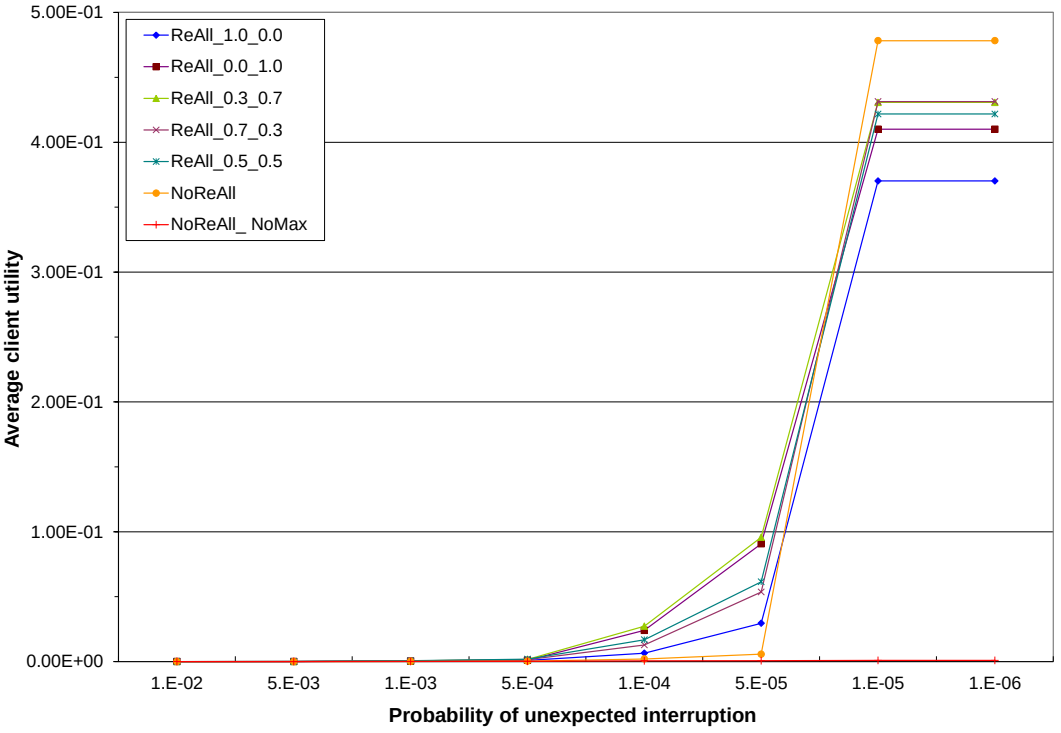
### 5.4.2 Re-allocating Tasks with the Different Priorities

In this section, we evaluate how the priorities for the two criteria, which are used to choose the best donor-task, might affect the client's utility for the different probabilities of task unexpected interruption. Here, we consider the different values of $W_1$ and $W_2$ for the SimTask re-allocation strategy "ReAll", which is compared to the cases when this strategy is not used, i.e. "NoReAll" and "NoReAll_NoMax". Here, the case "NoReAll_NoMax" also does not consider that a client can start its planned negotiation at the maximum of resource availability. The values of $W_1$ and $W_2$ are indicated in Figure 5.12 as the numbers on the label "ReAll" e.g., "ReAll_0.0_1.0" denotes that $W_1 = 0.0$ and $W_2 = 1.0$.

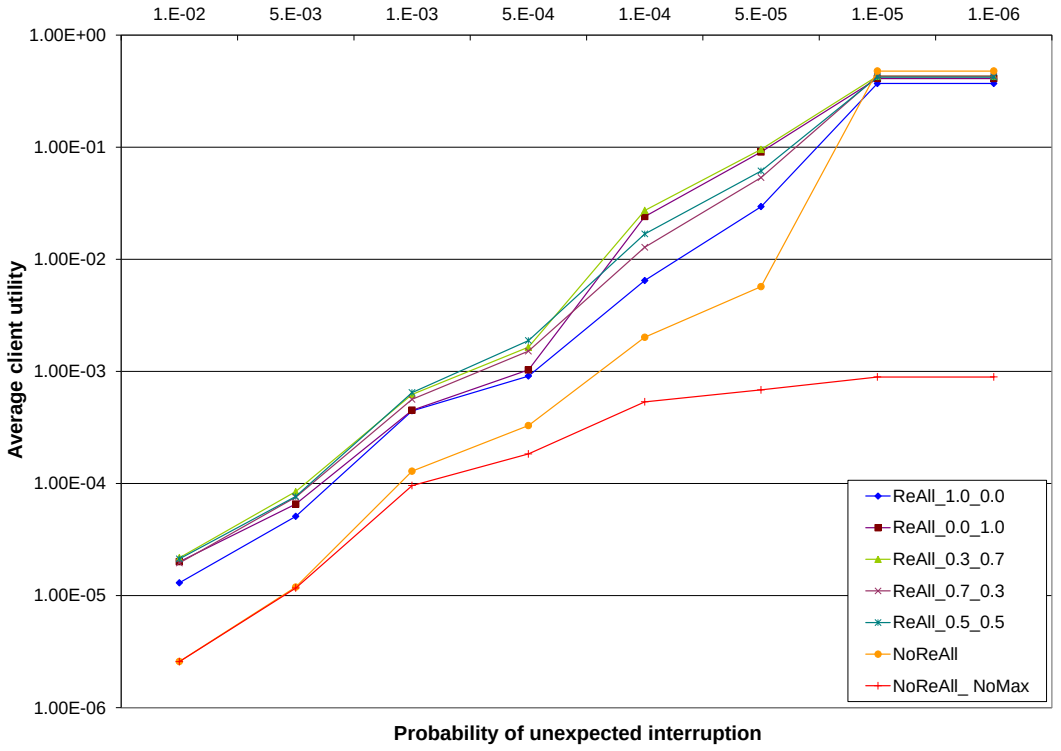
Generally, the utilities for the case $W_1 = 0.3$ and $W_2 = 0.7$ are larger than for all other combinations of the weight coefficients. It also has to be noted that the strict prioritising of the remaining allocation period of a donor candidate over its possible impact on other tasks, i.e. "ReAll_0.1_0.0", shows the smallest utilities among all combinations of the criteria weights. In the case, when the donor candidate's impact on execution of other tasks is strictly prioritised over the remaining allocation period, i.e. "ReAll_0.0_0.1", it also shows almost the smallest utilities for the larger probabilities of task unexpected interruption (above 5.E-04), while it demonstrates almost the largest utilities for the smaller probabilities (below 5.E-04). Here, we conclude that in our settings it is more beneficial for a client to prioritise the criterion $Cr_2^{j,i}(t_y, t)$ a bit more over the criterion $Cr_1^{j,l}(t)$, i.e. "ReAll_0.3_0.7". It is also important to mention that the difference in utilities is not large for the cases where the weights are more balanced such as "ReAll_0.3_0.7", "ReAll_0.7_0.3" and "ReAll_0.5_0.5", while "ReAll_0.3_0.7" usually shows the better utilities. This evaluation also demonstrates that the SimTask re-allocation strategy with any weights' combination outperforms almost all cases where it is not used such as "NoReAll" and "NoReAll_NoMax".

## 5.5 Conclusions

In this chapter, we have described a model of continuous inter-dependent tasks, where some tasks depend on the data from other tasks in order to be able to run. At the same time, the tasks which send data to other tasks are considered to be stopped, if the corresponding recipient-task is interrupted. If a recipient-task does not receive




(a) The average client utility over the probability of an unexpected interruption




(b) The average client utility in a logarithmic scale over the probability of an unexpected interruption

FIGURE 5.12: The changes in the client utility with the different preferences criteria

data from some (all) of its corresponding sender-tasks for some time, it stops due to a substantial increase in its calculations' error. If one task is interrupted, it affects not only its own utility, but the whole sub-tree of tasks and all connected recipient-task(s) (if applicable) Therefore, a task interruption in these settings represents even more substantial problem for a client than in previous chapters, where tasks did not depend on each others' data.

Here, a new re-allocation strategy, SimTask, has been introduced which allows a client to re-allocate resources among its own tasks through negotiation with the GRA, if an ordinary negotiation for resources becomes too long and this threatens a significant decrease in the client's utility. This strategy includes a decision making function, which considers several criteria, to choose the best donor-task when it is necessary. These criteria consider the remaining allocation period of time of the donor candidate, its influence on other tasks in a tree, if this donor candidate is chosen as a donor, and its status of execution at the time of negotiation.

The evaluation results show that SimTask outperforms the case when this strategy is not used for almost all probabilities of task unexpected interruption with the different levels of accuracy in the client's estimation of the maximum of resource availability. However, this re-allocation strategy might be ineffective for the cases of the small probabilities of task unexpected interruption and when the tasks are more likely to be allocated resources through an ordinary negotiation with the GRA. The evaluation also shows an improvement in the client's utility for the SimTask, compared to the cases when it is not used, in almost all cases with any combination of weights for the two criteria estimating donor candidates (also more balanced weights' combinations, such as 0.3 and 0.7, are more favourable for a client than the strict ones, such as 1.0 and 0.0).

# Chapter 6

# Case Study

## 6.1 Introduction

Some research [11, 13, 86] investigates the dynamism of real Grid resource utilisation and demand (see Section 2.4.2), allowing other researchers model a realistic Grid. This research shows deterministic patterns in resource utilisation and resource demand over days and weeks, which can be more or less distinctive for different Grid environments. Not all Grid environments exhibit deterministic patterns in resource utilisation, but we focus here on those which show some pseudo-periodic patterns in resource availability fluctuation. In our work, an assumption is that a pseudo-periodicity exists in the resource availability dynamics in a Grid due to the periodic demand on resources, which is supported by the work of Li et al. [87]. In this case study, we focus on the resource utilisation patterns demonstrated in the work of Iosup et al. [11], concentrating on those patterns which are pseudo-periodic. A pseudo-periodic resource fluctuation can be predictable and, therefore, it allows us to develop algorithms to utilise this predictability with the best outcome for the client.

In our work, we model a realistic environment for continuous and simultaneous tasks, presented in Chapters 4 and 5 e.g., the monitoring of climate parameters in the building. The continuity of task execution denotes that it is desirable for a task to be run without interruptions, while the simultaneity denotes that the tasks are preferably run at the same time, because they depend on each others' outputs. To realistically evaluate our strategies, we create a model derived from existing research [205–207], which focuses on automatic control inside buildings (e.g. climate control) in order to ensure

comfortable conditions, energy saving, etc. For example, the work of Qiao et al. [207] introduces a multi-agent system in which 'local' agents analyse the data from sensors and from 'personal' agents which learn the residents' preferences, while a 'central' agent aggregates their decisions. The hierarchical inter-connections between these agents are reflected in our research as the data dependencies among a client's tasks. The model of environmental changes (i.e. temperature) is simulated based on temperature changes illustrated by Lacroix et al. [206] and other realistic assumptions e.g., an increase of external temperature which may affect the temperature level inside the building. For instance, the controlled temperature inside the building in the work of Lacroix et al. oscillates pseudo-periodically during a day and, therefore, it has been modelled as a sine-type function in our case study.

In this chapter, we describe a simulator of the fluctuations of resource availability which follows a pattern of pseudo-periodicity observed in real Grid resource utilisation and this is discussed in Section 6.2. This section also indicates the behaviour of the Grid Resource Allocator (GRA) in negotiation with the client, depending on the resource availability and the demand on resources. In Section 6.3, we discuss our scenario, where our negotiation and re-allocation strategies for continuous and simultaneous tasks can be applied, as well as the environmental modelling (e.g. the change of temperature in the building). Sections 6.4 and 6.5 demonstrate the evaluation results for our ConTask negotiation strategy (see Section 4.3) and SimTask re-allocation strategy (see Section 5.3), while our adaptive fuzzy-controlled negotiation (see Chapter 3) is evaluated as part of the client reasoning in the ConTask negotiation strategy. Finally, Section 6.6 concludes and summarises our evaluation. Section 6.3 includes a table of notations, which are introduced in Sections 6.2 and 6.3 (see Table 6.1).

## 6.2 Resource Dynamism Simulation

In this section, we describe our Grid resource dynamism model, which is derived from real-life observation of resource utilisation in a Grid. Here, we also discuss the behaviour of the GRA in negotiation with the client, assuming that its level of greediness as well as its reservation value depend on the resource availability and/or resource demand.

### 6.2.1 Patterns of Real-life Resource Utilisation

Iosup et al. [11, 208] introduce the *Grid Workload Archive*, which provides tools to collect and analyse Grid workload data, and to share it with the community. They also illustrate the fluctuations of resource utilisation in different Grid systems during one month, which is the busiest among the $2-5$ months of data collection [11, p.678]. Here, they show the data from several Grid systems such as *LHC Computing Grid (LCG)* [209], *Grid3* [210], *DAS-2* [211] and *DAS-2 Grid*. All those Grid systems provide resources for the different resource-intensive tasks in physics, biomedicine, etc. Particularly, we are interested in the resource utilisation in Grid3, DAS-2 and DAS-2 Grid, which show some pseudo-deterministic changes in resource consumption over time. Another depicted Grid system, LCG, does not show pseudo-periodic fluctuations in resource consumption, so that its resources are almost fully occupied for the majority of the month with some steep falls at the end of the month.

Observing the resource utilisation presented in the work of Iosup et al. [11], we identify several patterns which are used further in our model of resource utilisation. First, Grid3 and DAS-2 Grid show a pseudo-periodicity in resource consumption over the days, where the period is approximately equal to one day. For example, the resource consumption is lower at night than during the day and this dependence is observed for all other days. DAS-2 Grid demonstrates approximately equal peaks of resource consumption every day, while Grid3 generally shows smaller peaks for the beginning and the end of a week. Here, we conclude that one of the patterns is a pseudo-periodicity of resource consumption over the days, and the level of consumption may vary at the same (equal peaks) or at the different (different peaks) rates every day.

Second, DAS-2 and Grid3 generally show repetition (pseudo-periodicity) in resource utilisation over a week. For example, the resource consumption is generally smaller at the beginning of a week and at the weekend than during the working days. However, the two last weeks for Grid3 are less distinguished from each other, that is the resource consumption does not fall significantly during the weekends. Consequently, the other pattern denotes a pseudo-periodicity of resource consumption over weeks, where the weeks' fluctuations of resource consumption can be approximately repeated for each week. In this way, we have identified two pseudo-periodicities over days and over weeks, which are superimposed on each other.

Finally, there are random oscillations in resource consumption over time, which is observed for all cases, and is expected for open and highly dynamic environments.

### 6.2.2 Model of Resource Utilisation

In this section, we describe our case study's resource utilisation model, which approximates the real patterns of pseudo-periodicity discussed in Section 6.2.1.

#### 6.2.2.1 Total Amount of Resources

Assume that the total amount of resources in a Grid at time $t$ is $N_{tot}(t)$. We assume that $N_{tot}(t)$ may change over time as resources may join or leave the Grid. The resources may also fail or be removed for upgrade and, therefore, the total amount of resources may decrease and then increase again over time. We also assume that a local Grid has limited capacity $N_{tot_0}$, where the total amount of resources cannot exceed this limit, i.e. $N_{tot}(t) \leq N_{tot_0}$. This assumption allows us to avoid unnecessary complexities in modelling the amount of busy or free resources over time. Here, we discuss modelling of the decreases and subsequent increases of the total resource amount up to its limit $N_{tot_0}$, which is presented in Algorithm 6.7.

As an example, Figure 6.1 shows a possible sequence of decreases and increases of the total resource amount, considering $N_{tot_0} = 100.0$. This figure demonstrates three typical changes in the total resource amount, modelled in our work, depicted in the time intervals $[t_A, t_B]$, $[t_B, t_C]$ and $[t_C, t_D]$. All these time intervals have their bottom limits $N_{tot_1}(t)$ of $N_{tot}(t)$ generated randomly at different moments of time $t$ (see Algorithm 6.7), i.e. $N_{tot_1}(t_A) = 60.0$ for $[t_A, t_B]$, $N_{tot_1}(t_B) = 55.0$ for $[t_B, t_C]$ and $N_{tot_1}(t_C) = 65.0$ for $[t_C, t_D]$. The functions which describe the decrease $f_{dec}(t)$ up to $N_{tot_1}(t)$ and increase $f_{inc}(t)$ up to $N_{tot_0}$ of the total amount of resources are presented as the damping functions with the inflection points $t_{dec}(t)$ and $t_{inc}(t)$ respectively, which are calculated at the moment of time $t$. In this figure, $t_{dec}(t_A) = 20.0$ and $t_{inc}(t_A) = 60.0$ for the decrease and increase of $N_{tot}(t)$ in the time interval $[t_A, t_B]$, $t_{dec}(t_B) = 90.0$ and $t_{inc}(t_B) = 110.0$ for $[t_B, t_C]$, and $t_{dec}(t_B) = 105.0$ and $t_{inc}(t_B) = 130.0$ for $[t_C, t_D]$.

According to this example, the decrease of $N_{tot}(t)$ is modelled after the current time $t$ (e.g. $t_A < t_{dec}(t_A)$), where the closer time point $t_{dec}(t)$ is to the current time $t$, the more steep its fall towards $N_{tot_1}(t)$. An increase of $N_{tot}(t)$ might be delayed as in the

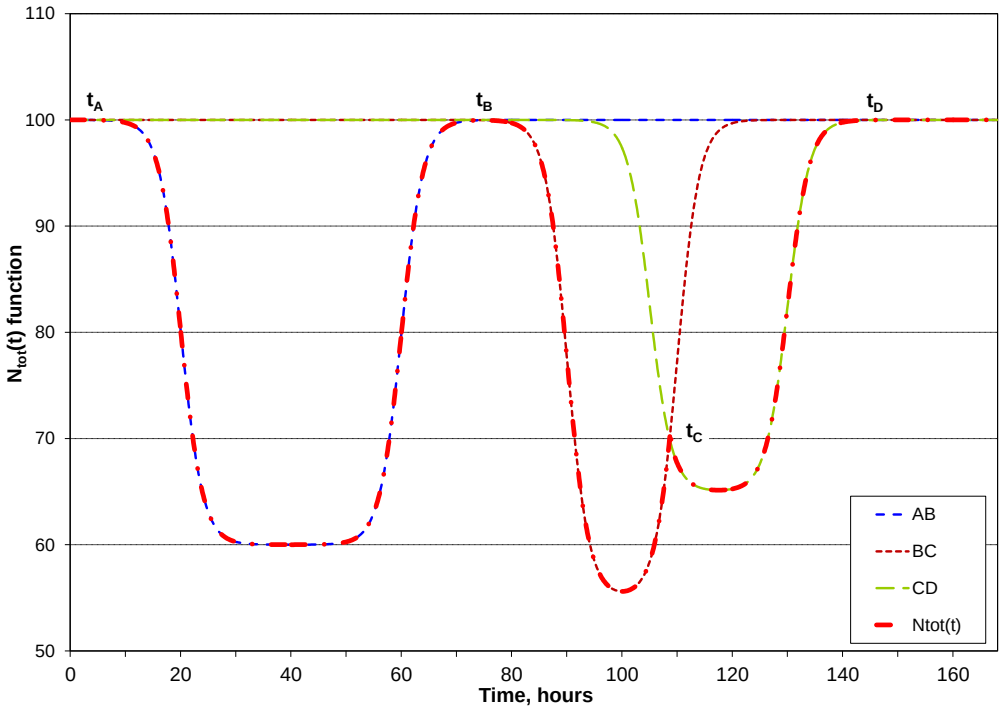



FIGURE 6.1: A fluctuation of the total amount of resources over time

time interval $[t_A, t_B]$ or start almost instantly after a decrease of $N_{tot}(t)$ as in the time interval $[t_B, t_C]$, and these patterns are determined by the difference between $t_{inc}(t)$ and $t_{dec}(t)$ at time $t$. In the time interval $[t_C, t_D]$, a decrease of $N_{tot}(t)$ occurs before $N_{tot}(t)$ reaches its upper limit $N_{tot_0}$, creating a local peak of the total resource amount where $t_C > t_{dec}(t_C)$, while the time intervals $[t_A, t_B]$ and $[t_B, t_C]$ show only local lows.

Consequently, the damping functions which reflect a decrease or increase of $N_{tot}(t)$ over time can be presented as in the following formula, where $\tau$ determine a speed of those decreases and increases:

$$\begin{aligned} f_{dec}(t) &= \frac{N_{tot_0} - N_{tot_1}(t)}{e^{(t-t_{dec}(t))/\tau} + 1} + N_{tot_1}(t), \\ f_{inc}(t) &= \frac{N_{tot_0} - N_{tot_1}(t)}{e^{(t_{inc}(t)-t)/\tau} + 1} + N_{tot_1}(t). \end{aligned} \tag{6.1}$$

Then $N_{tot}(t)$ is described formally as:

$$N_{tot}(t) = \frac{1}{N_{tot_1}(t)} \times f_{dec}(t) \times f_{inc}(t). \tag{6.2}$$

The algorithm for calculating the total resource amount fluctuation is presented below in Algorithm 6.7. Here, the total resource amount $N_{tot}(t)$ may decrease from a value

**Algorithm 6.7** An algorithm to calculate $N_{tot}(t)$

1: **repeat**
2: Each virtual second $t$
3: {A decrease in $N_{tot}(t) \approx N_{tot_0}$ occurs randomly $X_{dec}$ with the probability $Prob_{dec}$ e.g., $t_A$ or $t_B$ in Figure 6.2}
4: **if** $(X_{dec} < Prob_{dec})\&(N_{tot_0} - N_{tot}(t) < Var_{dec})$ **then**
5: Calculate $t_{dec}(t) = t + \tau_{dec}(t)$
6: Calculate $t_{inc}(t) = t_{dec}(t) + \tau_{inc}(t)$
7: Generate randomly $N_{tot_1}(t) < N_{tot_0}$
8: **end if**
9: Calculate $N_{tot}(t)$ (see Equation (6.2))
10: {A decrease in $N_{tot}(t)$ after its 'fall' from $N_{tot_0}$ occurs randomly $X'_{dec}$ with the probability $Prob'_{dec}$ e.g., $t_C$ in Figure 6.1}
11: **if** $(X'_{dec} < Prob'_{dec})\&(N_{tot_0} - N_{tot}(t) < Var'_{dec})\&(dN_{tot}(t) > 0)$ **then**
12: Re-generate randomly $N_{tot_1}(t) < N_{tot}(t)$
13: Calculate $t_{dec}(t) = t + \tau'_{dec}(t)$
14: Calculate $t_{inc}(t) = t_{dec}(t) + \tau'_{inc}(t)$
15: **end if**
16: **until** $t_{dl}^{exec}$

close or equal to its upper limit (see lines $3-9$) e.g., $N_{tot_0} - N_{tot}(t)$ is smaller than a chosen experimentally coefficient $Var_{dec} = 0.001$. It may also decrease again (i.e. a 'secondary' decrease) when its value has not reached an upper limit yet (see lines $10-16$), i.e. $N_{tot_0} - N_{tot}(t)$ is smaller than a chosen coefficient $Var'_{dec} = 15.0$. Both cases of decrease with subsequent increase occur randomly, according to the probabilities $Prob_{dec}$ and $Prob'_{dec}$ respectively. The 'secondary' decrease occurs when the total resource amount is in a process of increasing towards $N_{tot_0}$, i.e. $dN_{tot}(t) > 0$, which allows us to simulate local peaks as depicted in Figure 6.2. The time functions $\tau_{dec}(t)$, $\tau_{inc}(t)$, $\tau'_{dec}(t)$ and $\tau'_{inc}(t)$ generate random values from experimentally chosen time intervals. That is, $\tau_{dec}(t)$ produces values from the interval $[0, 10 \times \tau]$ virtual hours, $\tau_{inc}(t)$ and $\tau'_{inc}(t)$ produce values from the interval $[0, 5]$ virtual days, and $\tau'_{dec}(t)$

produces values from the interval $[-5 \times \tau, 5 \times \tau]$ virtual hours, converted into virtual seconds with a uniform distribution.

#### 6.2.2.2 Busy and Free Resources

The total amount of resources consists of the amounts of busy, $N_b(t)$, and free, $N_f(t)$, resources at time $t$. The amount of busy resources fluctuates according to a pseudo-periodic pattern, and is naturally affected by the total amount of resources as well as the amount of requested resources. Figure 6.2 shows an example of resource fluctuations over time and how the change in the total amount of resources affects the amounts of free and busy resources. When the total amount of resources $N_{tot}(t)$ decreases, the amounts of free $N_f(t)$ and busy $N_b(t)$ resources decrease as well and vice versa. Also note that every local maximum of free resources corresponds to a local minimum of busy resources.

In our model, resources are released and allocated continuously and, therefore, their amounts are larger than zero at any time but can be in close proximity to zero. This model reflects the resource utilisation depicted in Figure 6.1, as almost all investigated Grid systems do not show an utilisation of all resources at once (except LCG). This also complies with the GRA's behaviour, discussed in Chapters 3-5, when interrupted tasks are not re-allocated immediately because the GRA has to negotiate their re-allocation with a client. That is, if the total resource amount decreases (due to resource failure or withdrawal), the interrupted tasks do not automatically consume the rest of the free resource amounts and, hence, some resources remain available. The total resource amount in terms of busy and free resource amounts is presented in the following equation:

$$N_{tot}(t) = N_b(t) + N_f(t). \tag{6.3}$$

In our simulator, resources can be in either of two states *free* and *busy*, where the first denotes that a resource is available and the latter denotes that it is allocated to a particular task. Here, the change in resource availability mainly depends on the change in the level of demand on resources, because the amount of requested resources affects the amount of resources transferred from a free to busy states per time unit. The level of demand at time $t$ is described by a function $N_d(t)$, and this level in the period of

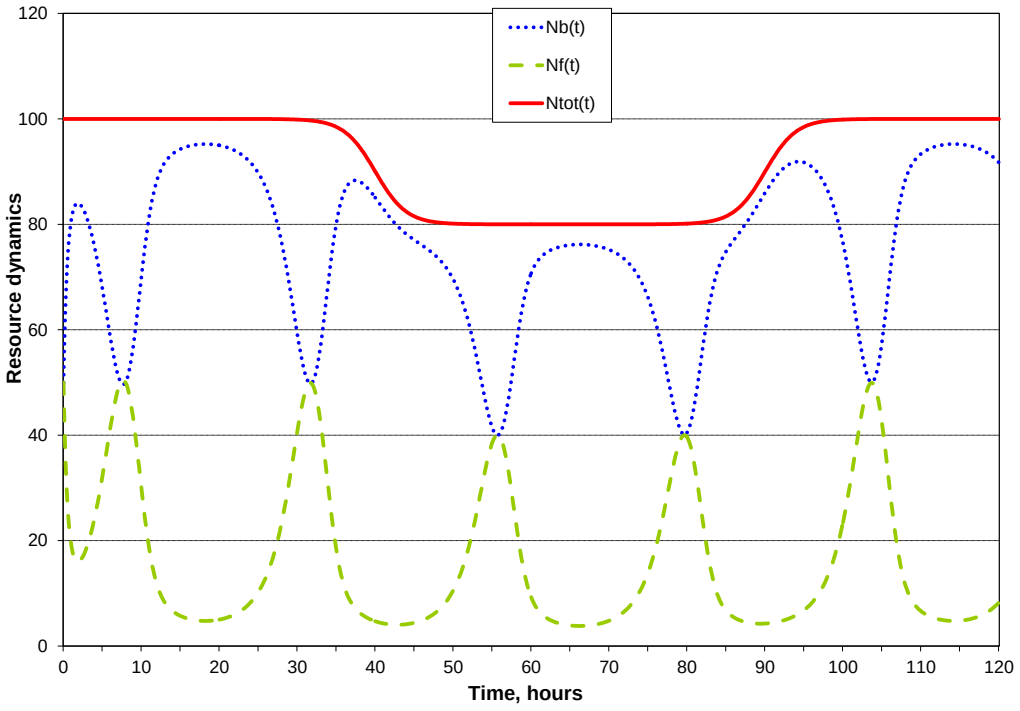



FIGURE 6.2: An example of resource fluctuations in a Grid

time $\Delta t$ is depicted as in the following equation:

$$N_d\left(t+\Delta t\right)=N_d\left(t\right)+\Delta N_d\left(t\right), \tag{6.4}$$

where $\Delta N_d\left(t\right)$ shows the change in the level of demand during $\Delta t$, i.e. the amount of requested resources which are still waiting for allocation. Now, we define the speed of resource transition from a busy state into a free state and back. This speed defines the portion of free or busy resources which become busy or free per time unit. In the case when resources are transferred from a busy to free state, it means that they have been released by the tasks. The speed of this transition $\beta_0 N_b\left(t\right)$ is determined by the amount of busy resources which are transferred into free state per time unit. In the case, when resources are transferred from a free to busy state, it means that they have been allocated to the tasks, according to the amount requested, $N_d\left(t\right)$. The speed of this transition $\alpha_0 N_f\left(t\right) N_d\left(t\right)$ is determined by the amount of free resources which are transferred into busy state due to clients' requests. In our current model, the coefficients $\alpha_0$ (1/([time unit]*[resource unit])) and $\beta_0$ (1/[time unit]) are chosen

to be constants of resource fluctuations in a Grid in order to make our model more deterministic in terms of analysis. These constants are chosen experimentally to ensure that the busy resource fluctuations follow the observations of real resource utilisation described in Section 6.2.1.

Considering the demand function $N_d(t)$, we have to derive an equation which enables us to calculate the amount of free or busy resources at time $t$. The amount of busy resources in the $\Delta t$ period of time is:

$$N_b(t+\Delta t) = N_b(t) + \Delta N_b(t), \tag{6.5}$$

where $\Delta N_b(t)$ shows the change in the amount of busy resources during $\Delta t$ and $N_b(t+\Delta t)$ is presented in the following equation:

$$N_b(t+\Delta t) = N_b(t) - \beta_0 N_b(t)\Delta t + \alpha_0 N_f(t) N_d(t)\Delta t + \upsilon(t)\Delta N_{tot}(t), \tag{6.6}$$

where $\beta_0 \times N_b(t) \times \Delta t$ is the amount of resources which becomes free during $\Delta t$, $\alpha_0 \times N_f(t) \times N_d(t) \times \Delta t$ is the amount of resources which becomes busy during $\Delta t$ and $\upsilon(t) \times \Delta N_{tot}(t)$ is the change in the total amount of resources during $\Delta t$, part of which $\upsilon(t) \in [0,1]$ was busy resources. For example, the total amount of resources may decrease, because some resources fail or leave a Grid and some proportion of these resources can be busy at the time of failure or withdrawal.

The amount of free resources in $\Delta t$ period of time is:

$$N_f(t+\Delta t) = N_f(t) + \Delta N_f(t), \tag{6.7}$$

where $\Delta N_f(t)$ is the change in the amount of free resources during $\Delta t$ and this amount is calculated as in the following equation:

$$N_f(t+\Delta t) = N_f(t) + \beta_0 N_b(t)\Delta t - \alpha_0 N_f(t) N_d(t)\Delta t + \mu(t)\Delta N_{tot}(t), \tag{6.8}$$

where $\mu(t) \in [0,1]$ is the portion of the total resource amount changes which was free at the time of change. The sum of values $\mu(t)$ and $\upsilon(t)$ is equal to 1, that is each of these values defines a proportion of free or busy resources which left or joint a Grid by changing the total amount of resources in a Grid. In the case, when the total amount increases $\Delta N_{tot}(t) > 0$, $\upsilon(t) = 0$ and $\mu(t) = 1$. That is, the total amount of resources increases only in terms of the additional free resources. However, when the

total amount decreases $\Delta N_{tot}(t) < 0$, then $\upsilon(t) \in [0,1]$ and $\mu(t) \in [0,1]$ but their sum $\upsilon(t) + \mu(t) = 1$. In other words, the total amount of resources decreases in terms of the current busy and free resources.

#### 6.2.2.3 Demand on Resources

The demand on resources intuitively decreases when some jobs are allocated the resources or the total amount of resources increases. However, the demand increases when some tasks are interrupted or new tasks enter the Grid and request resources. The level of demand on resources has a pseudo-periodic fluctuations, which result in a pseudo-periodicity in the changes of the busy and free amounts of resources over time. Hence, the resource demand function has a periodic component $\gamma(t)$, which denotes the periodicity in the increase or decrease of the amount of newly requested resources over time. For example, the increase in new requests may happen during the day-time when the clients launch their tasks, but it may decrease at night when the tasks finish their execution. Such periodicity can be also observed through a week e.g., the usage of resources decreases during weekends and increases during working days. Consequently, the periodic component $\gamma(t)$ is described in the following equation:

$$\gamma(t) = \gamma_0 \left(1 - \gamma_1 \times \sin\left(\frac{2\pi t}{T_{dyn}}\right)\right)^{deg_1} \times \left(\gamma_2 + \left|\sin\left(\frac{2\pi t}{14 \times T_{dyn}}\right)\right|^{deg_2}\right), \tag{6.9}$$

where $\gamma_0$, $\gamma_1$ and $\gamma_2$ are experimentally chosen coefficients, i.e. $\gamma_0 = 11$ ([resource unit]/[time unit]), $\gamma_1 = 0.9$ and $\gamma_2 = 0.1$ [no dimension]; $T_{dyn}$ is the period of a sine-type wave and is equal to 24 virtual hours; $deg_1$ and $deg_2 \in [0,1]$ are the degrees which define the level of distinction between the peaks of newly requested resource amounts over days and weeks, and $t$ is the current time. The function of newly requested resource amounts $\gamma(t)$, presented in Equation (6.9), shows two inter-connected periodicities as the sine-type waves, where the first factor denotes a periodicity over day-time with a period of 24 virtual hours, while the second factor denotes a periodicity over weeks with a period of one virtual week. Here, smaller values of $deg_1$ and $deg_2$ lead to less deterministic periodicities among days and weeks e.g., as a result, approximately the same level of resource utilisation can be observed for two days without distinctive peaks for each day. Less determinism for a week periodicity in the newly requested resource amounts means that the resource utilisation might not significantly decrease over the weekends.

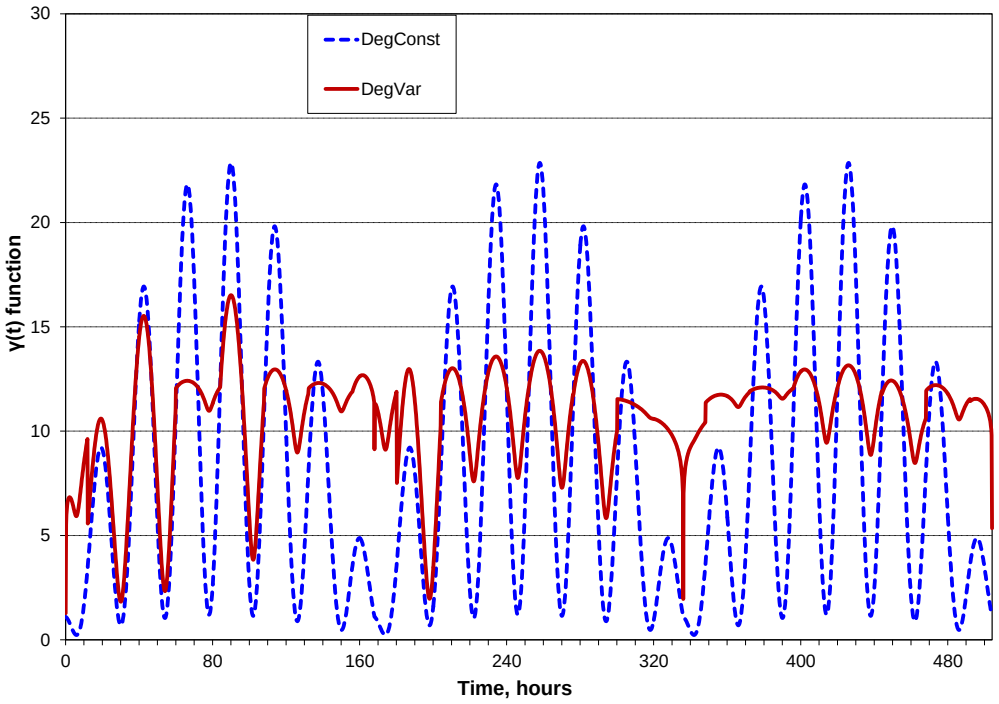



FIGURE 6.3: A simulation of $\gamma(t)$ for degrees constantly equal to 1, 'DegConst', and for varying degrees, 'DegVar'

The function $\gamma(t)$ of newly requested resource amounts per time unit $t$ with the different configurations of $deg_1$ and $deg_2$ is depicted in Figure 6.3. In the 'DegConst' case, the degrees $deg_1$ and $deg_2$ are constants and equal to one, which denotes the high level of distinction between the peaks of newly requested resource amounts over days and weeks, as opposed to the 'DegVar' case, where $deg_1$ and $deg_2$ are re-calculated randomly in the interval $[0, 1]$ once every 24 virtual hours. This figure shows that when the degrees are smaller than one, there is less distinction between the peaks of newly requested resources over days and less similarity between the series of peaks over weeks, which means less determinism in terms of periodic patterns for a client.

Finally, the change in demand on resources in the period of time $\Delta t$, $N_d(t + \Delta t)$ can be presented as in the following equation:

$$N_d(t + \Delta t) = N_d(t) - \alpha_0 N_f(t) N_d(t) \Delta t + \gamma(t) \Delta t - \upsilon(t) \Delta N_{tot}(t), \tag{6.10}$$

where the component $\alpha_0 N_f(t) N_d(t) \Delta t$ decreases a demand on resources, because some of the requested resources have been allocated; $\gamma(t) \Delta t$ increases a demand on resources, because of the newly requested resources, and $\upsilon(t) \Delta N_{tot}(t)$ increases a demand on resources, because of decrease in the total amount of resources.

#### 6.2.2.4 Final Solution

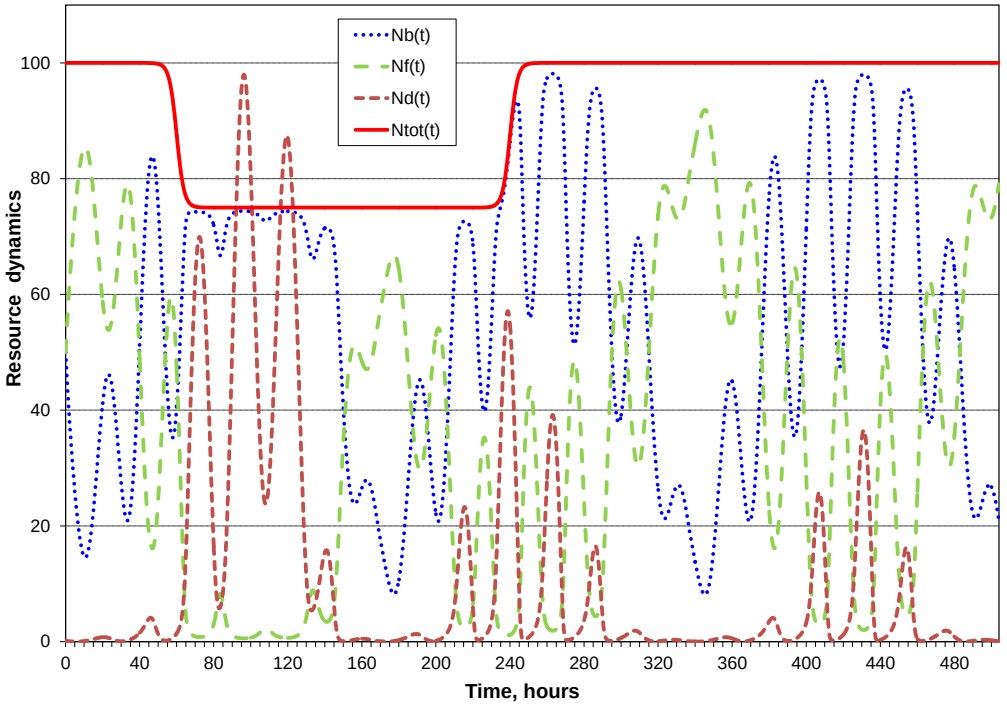
To calculate the amounts of busy and free resources as well as the demand on resources, we solve Equations (6.6), (6.8) and (6.10) by considering Equation (6.3) and assuming $\Delta t \to 0$, as below:

$$\begin{cases} \dfrac{dN_f(t)}{dt} = -(\alpha_0 N_d(t) + \beta_0) N_f(t) + \beta_0 N_{tot}(t) + \mu(t) \dfrac{dN_{tot}(t)}{dt}, \\ \dfrac{dN_d(t)}{dt} = -\alpha_0 N_f(t) N_d(t) + \gamma(t) - \upsilon(t) \dfrac{dN_{tot}(t)}{dt}. \end{cases} \tag{6.11}$$

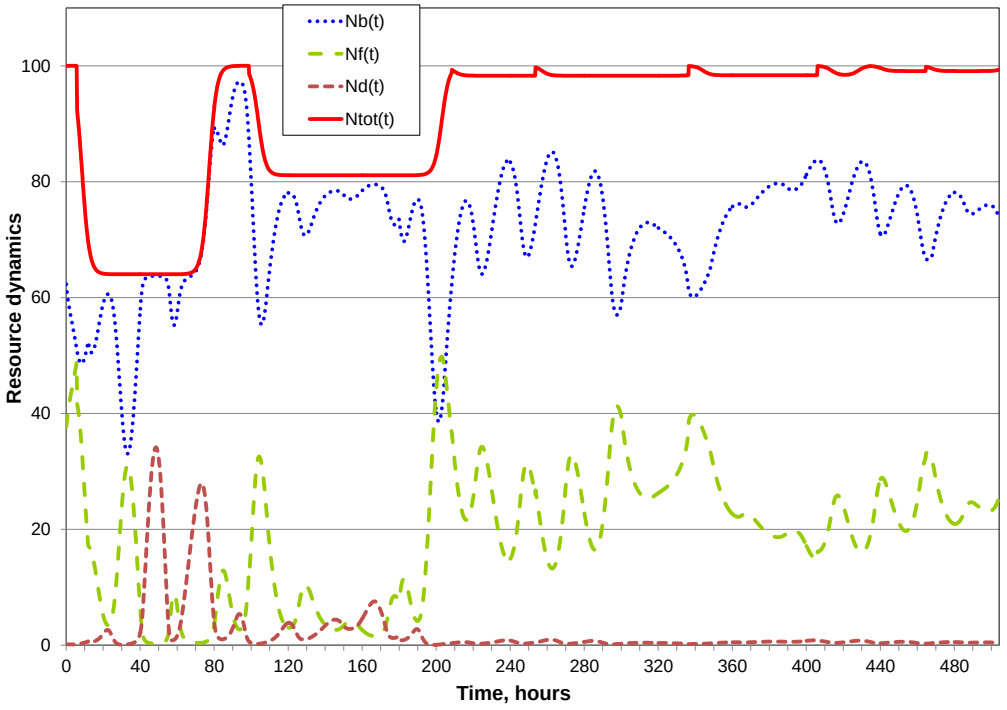
As long as $N_b(t + \Delta t)$ and $N_f(t + \Delta t)$ are linearly dependent functions, then we choose only one of them to solve the system of differential equations (6.6) and (6.8).

Figure 6.4 shows a simulation of resource fluctuations for three weeks, modelled as described in this section, with constant and varying degrees of $\gamma(t)$. Figure 6.4(a) demonstrates Grid resource availability and demand changes over time with constant degrees of determinism in the daily $deg_1 = 1.0$ and weekly $deg_2 = 1.0$ periodicities. In this figure, a distinction among the day peaks and among the week endings is clear and it repeats over time, except for the case when the day peaks are affected by the decrease in the total amount of resources. It also has to be noted that the decrease in the total amount of resources together with the high resource scarcity leads to a significant increase in the total demand on resources, because the less number of requests can be satisfied. Generally, larger demand reflects larger resource scarcity, fluctuating pseudo-periodically over time.

Figure 6.4(b) shows our simulation of resource utilisation, considering the degrees of determinism $deg_1$ and $deg_2$ can vary over days in the interval $[0, 1]$. This is a simulation of resource fluctuations which is used in a further evaluation of our ConTask negotiation strategy and SimTask re-allocation strategy, while the previous figures in this section depict the simplified and/or idealistic examples of resource fluctuations in order to explain some aspects of our model. In this figure, the periodicities among the days and weeks, compared to Figure 6.4(a), are much less noticeable. For example, the amount




(a) An example of resource fluctuations in a Grid with $deg_1 = 1.0$ and $deg_2 = 1.0$ for all days




(b) An example of resource fluctuations in a Grid with varying $deg_1$ and $deg_2$

FIGURE 6.4: A demonstration of periodicity in resource fluctuations

of busy resources does not decrease substantially for the weekends between the first and the second weeks (i.e. in the time interval of $[120, 168]$ hours), but it decreases more deterministically for the weekends between the second and the third weeks (i.e. in the time interval of $[228, 336]$ hours). It also has to be noted that the peaks of resource consumption among the days are less distinctive when the total amount of resources significantly decreases, while it is more distinctive for other cases.

### 6.2.3 GRA's Negotiation Behaviour

We assume that the change of the GRA's reservation value depends on the change in resource availability in the Grid e.g., a decrease in resource availability leads to a decrease in this reservation value. Previously (see Chapters 3-5), we did not focus on the modelling of resource availability fluctuations explicitly, but it has been considered to be reflected implicitly in the changes of the GRA's reservation value. In this chapter, we have developed a continuous Grid resource simulator discussed above which allows us to model different GRA's behaviours, depending on different influential factors such as the resource demand and availability explicitly.

In this and previous chapters, the reservation value of the GRA is calculated as some proportion of a maximum value of a client $\tau_{i,l}^{max}$ (see Definition 4.7). Hence, the GRA's reservation value is always smaller or equal to the client's maximum value (see Section 3.2.2.3), i.e. the GRA will not offer a client an amount of resources larger than the client's initial request. In this chapter, the GRA's reservation value $G_{i,j,l}^{max}(t)$ has a periodic component $Per(t)$, which denotes a ratio of resource availability in respect of the total resource amount's limit $N_{tot_0}$ at time $t$. It also has a random component $Random_{i,j,l}$ (see Equation (4.5)), which models the GRA's individual decision-making process in respect of a client for each task $i$ at round $j$ during an interruption period $\tau_{i,l}^{int}$. A periodic component $Per(t)$ is presented in the following equation:

$$Per(t) = N_f(t) / N_{tot_0}, \tag{6.12}$$

where $Per(t) \in [0, 1]$ at time $t$. A random component simulates the GRA's behaviour where the GRA's reservation value may have larger or smaller fluctuations, depending on the coefficient $K_3$ (see Equation (4.5)). This coefficient is the percentage of the client's maximum value $\tau_{i,l}^{max}$, which determines a standard deviation of the random component of $G_{i,j,l}^{max}(t)$. If $K_3 = 0\%$, it means no random fluctuations, while $K_3 = 100\%$

denotes that the standard deviation of the GRA's reservation value is equal to the client's maximum value. That is, the larger $K_3$, the larger are oscillations of $G^{max}_{i,j,l}(t)$ over time. Finally, the GRA's reservation value is calculated as below:

$$G^{max}_{i,j,l}(t) = \tau^{max}_{i,l} \times Per(t) + Random_{i,j,l}, \tag{6.13}$$

where the larger amount of available resources leads to the GRA's reservation value being closer to the client's maximum value, and $Random_{i,j,l}$ produces a random percentage of $\tau^{max}_{i,l}$ with the standard deviation, determined by $K_3$.

It has to be noted that not only the GRA's reservation value is sensitive to the changes in a Grid environment, but also its level of greediness $\beta^{gra}_{i,l}(t)$. That is, the GRA becomes more greedy if its reservation value decreases (as a result of a decrease in resource availability) and a demand on resources is larger than its availability, and vice versa. If the GRA's reservation value decreases or increases and the demand on resources is smaller or larger than its availability respectively, then the GRA's level of greediness remains the same as at previous time step.

## 6.3 Environmental Changes Simulation

In this section, we describe how we model the environmental changes, i.e. the level of temperature, which is monitored by continuous and simultaneous tasks. In Chapter 5, we introduced a parameter $P_{i,S_i}(t)$ (see Definition 5.2) monitored over $t$ by task $i$ which has a set of sender-tasks $S_i$ (this set is empty for the lowest layer tasks). This parameter is directly estimated by the lowest layer tasks, while the higher layer tasks calculate its value as a linear combination of the values of this parameter, received from their corresponding sender-tasks. In this chapter, we model the level of temperature which corresponds to this parameter in Chapter 5. Here, we assume that the level of temperature changes pseudo-periodically over time as demonstrated in the work of Lacroix et al. [206, p.7] if a heater is working in a room. In our simulation, the objective of a heater is to maintain the preferable temperature level $T_{pref}$ in all rooms in a building. For example, when the level of temperature falls below this preferable level, then an actuator initiates the start of heating. If this temperature level rises till the preferable level, then the heating is stopped but the remaining heat keeps increasing the temperature level for some time, i.e. the temperature level may increase above the preferable one. Such periodicity for internal building temperature $T_{int}$ is

depicted by Lacroix et al. [206, p.7]. The authors also depict the temperature level outside the building, which is $T_{ext}$. There are also other temperatures, related to the water heater e.g., the storage tank temperatures at the top ($T_{wht}$), middle ($T_{whm}$) and bottom ($T_{whb}$). The other temperature, presented by the authors, $T_{sol}$ is from the solar collector.

The temperature inside a room can be affected by the different factors such as the external temperature outside of the room / building, etc. In the figure depicted by Lacroix et al. [206, p.7], the temperature also changes periodically over a period of approximate half an hour ($T_{per}^{tem}$) when the heater is working. Consequently, in our model of temperature changes, we also simulate its fluctuations with a period of half an hour, and it normally fluctuates between 18 and 22°C as depicted in Figure 6.5, where 20°C is considered to be the preferable temperature level in the room. We also model such cases when the level of temperature can increase or decrease because of external factors (e.g. the input temperature from the outside of the building), which may cause the non-stationary fluctuations of the temperature (increase or decrease). For example, an abnormal increase in the temperature as depicted in Figure 6.5, which might occur due to the rise of the external temperature if a room has only heating but no cooling system, which is common case for ordinary residential buildings. A non-stationary decrease in the temperature level may occur as a result of open windows and doors, which cause a transition of cool air inside the building.

In the case of the non-stationary temperature deviations, the deviated temperature from the preferable one $T_{pref}$ is modelled as the function $T_{nonst_i}(t)$ over time $t$ for task $i$. A deviation is calculated with the damping functions $f'_{dec_i}(t)$ and $f'_{inc_i}(t)$ similar to the calculation of the total amount of resources, presented in Section 6.2.2.1. As opposed to Section 6.2.2.1, the temperature can increase above the preferable temperature level, i.e. it may decrease or increase up to $T_{bor_i}(t)$, where $T_{bor_i}(t)$ can be larger or smaller than $T_{pref}$. It is also modelled that one fluctuation in the temperature does not occur simultaneously with another one, i.e. it may only occur when the current temperature is approximately equal to $T_{pref}$. The functions $f'_{dec_i}(t)$ and $f'_{inc_i}(t)$ are presented in Equation (6.14).

$$
\begin{aligned}
f'_{dec_i}(t) &= \frac{T_{pref} - T_{bor_i}(t)}{e^{\left(t - t'_{dec_i}(t)\right)/\tau_i^{dc}(t)} + 1} + T_{bor_i}(t)\,, \\
f'_{inc_i}(t) &= \frac{T_{pref} - T_{bor_i}(t)}{e^{\left(t'_{inc_i}(t) - t\right)/\tau_i^{ic}(t)} + 1} + T_{bor_i}(t)\,,
\end{aligned}
\tag{6.14}
$$

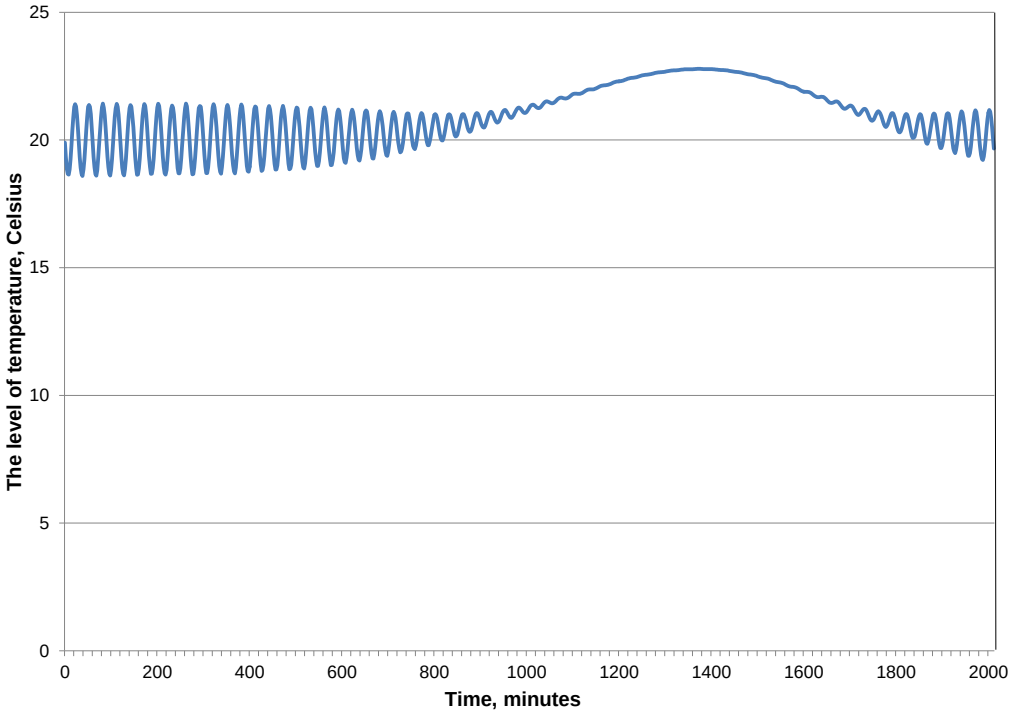



FIGURE 6.5: The change in temperature level over time

where $t^{'}_{dec_i}(t)$ and $t^{'}_{inc_i}(t)$ are the inflection points for $f^{'}_{dec_i}(t)$ and $f^{'}_{inc_i}(t)$ respectively, while the values $\tau^{dc}_{i}(t)$ and $\tau^{ic}_{i}(t)$ determine the speed of decrease or increase of the functions $f^{'}_{dec_i}(t)$ and $f^{'}_{inc_i}(t)$ at time $t$ for task $i$ respectively. The deviated temperature $T_{nonst_i}(t)$ at the moment of time $t$ is calculated as in Formula (6.15).

$$T_{nonst_i}(t) = 0.5 \times \left( T_{pref} + \frac{1}{T_{bor_i}(t)} \times f^{'}_{dec_i}(t) \times f^{'}_{inc_i}(t) \right). \tag{6.15}$$

In addition to the non-periodic temperature deviations, we also model slight periodic temperature oscillations, which describe the heating process depicted in the figure by Lacroix et al. [206, p.7], around $T_{pref}$ or $T_{nonst_i}(t)$. In this figure, the temperature level fluctuates around the preferable one for the internal temperature in the building. We imitate such temperature oscillations by calculating the amplitude of these oscillations $T_{amp_i}(t)$ for each task $i$ as a sum of: first, a task-independent temperature amplitude component $T_A$ (e.g. $T_A = 2°\text{C}$); second, a task-dependent $i$ temperature amplitude component $T_{A_i}$ and third, a time $t$ and task-dependent $i$ temperature amplitude component $T_{A_{i,t}}$, and this sum is multiplied on the value of function $f_{A_i}(t)$, which can

attenuate this amplitude in the case of significant non-stationary temperature changes. Here, the attenuation function $f_{A_i}(t)$ suppress the temperature level oscillations, when the heater is not working or unable to significantly affect the temperature in a room, as depicted in Figure 6.5 in the time interval, where the temperature level rises significantly above the preferable one.

The temperature amplitude component $T_{A_i}$ is calculated randomly from the interval $[-T_A, +T_A]$ once for each task, while $T_{A_{i,t}}$ is calculated randomly from the interval $[-T_A/20, +T_A/20]$ for each task $i$ every time step $t$. Here, the component $T_{A_i}$ can more significantly affect the temperature oscillation amplitude than the component $T_{A_{i,t}}$. We assume that the rooms in the building can be of the different sizes, thermal isolation, etc. and, therefore, they should have distinctive temperature deviations $T_{A_i}$, while those deviations $T_{A_{i,t}}$ should not normally be significant for a particular room over time. Consequently, the final amplitude of those small temperature oscillations is calculated as in the following equation:

$$T_{amp_i}(t) = 0.5 \times \left(T_A + T_{A_i} + T_{A_{i,t}}\right) \times f_{A_i}(t), \tag{6.16}$$

where the attenuation function is presented as:

$$f_{A_i}(t) = e^{-k_A \times \left|1.0 - ((2 \times T_{nonst_i}(t) - T_{pref})/T_{pref})\right|}, \tag{6.17}$$

which is equal to one when there is no non-stationary temperature deviations, i.e. $T_{nonst_i}(t) = T_{pref}$, while $k_A$ is an empirically chosen coefficient which is equal to 40 in our model. As long as the temperature level oscillations are pseudo-periodic, we model this pseudo-periodicity as a sine-type function with the period $T_{per}^{tem}$, derived from figure depicted in the work [206, p.7], and the phase $Ph_i$ for each task $i$. Finally, the temperature level $T_i(t)$ estimated by the lowest layer task $Layer_i = 0$ (see Section 5.2.1) at time $t$ is calculated as in the following equation:

$$T_i(t) = T_{nonst_i}(t) - T_{amp_i}(t) \times \sin\left(2\pi\left(t - Ph_i\right)/T_{per}^{tem}\right). \tag{6.18}$$

TABLE 6.1: List of notation for Sections 6.2 and 6.3

| Symbol | Notation |
|---|---|

Continued on the next page

Continued from the previous page

| **Symbol** | **Notation** |
|---|---|
| $N_{tot}(t)$ | A total amount of resources in a Grid at time $t \in \mathbb{R}$, where $N_{tot}(t) > 0$, $N_{tot}(t) \in \mathbb{R}$. |
| $N_{tot_0}$ | An upper limit of the total amount of resources in a Grid, where $N_{tot_0} > 0$. |
| $N_{tot_1}(t)$ | A bottom limit of the total amount of resources at time $t \in \mathbb{R}$, where $N_{tot_1}(t) > 0$. |
| $f_{dec}(t)$ | A damping function which models a decrease in the total resource amount up to $N_{tot_1}(t)$ with an inflection point $t_{dec}(t)$ calculated at time $t \in \mathbb{R}$, where $t_{dec}(t) \in \mathbb{R}$. |
| $f_{inc}(t)$ | A damping function which models an increase in the total resource amount up to $N_{tot_0}$ with an inflection point $t_{inc}(t)$ calculated at time $t \in \mathbb{R}$, where $t_{inc}(t) \in \mathbb{R}$. |
| $\tau$ | A value which determines a speed of decrease or increase in the total resource amount, where $\tau > 0$, $\tau \in \mathbb{R}$. |
| $\tau_{dec}(t)$, $\tau_{inc}(t)$, $\tau'_{dec}(t)$, $\tau'_{inc}(t)$ | The functions which denote the randomly generated intervals of time, used to model the inflection points $t_{dec}(t)$ and $t_{inc}(t)$ at time $t \in \mathbb{R}$, where $t_{dec}(t), t_{inc}(t) \in \mathbb{R}$. |
| $N_b(t)$ | An amount of busy resources at time $t \in \mathbb{R}$, where $N_b(t) \geq 0$, $N_b(t) \in \mathbb{R}$. |
| $N_f(t)$ | An amount of free resources at time $t \in \mathbb{R}$, where $N_f(t) \geq 0$, $N_f(t) \in \mathbb{R}$. |
| $N_d(t)$ | A total amount of requested resources at time $t \in \mathbb{R}$, where $N_d(t) \geq 0$, $N_d(t) \in \mathbb{R}$. |
| $\alpha_0, \beta_0$ | The coefficients which determine the speed of resource transition from a busy to free and vice versa states respectively, where $\alpha_0 > 0$, $\beta_0 > 0$. |

Continued on the next page

Continued from the previous page

| **Symbol** | **Notation** |
|---|---|
| $\upsilon(t)$, $\mu(t)$ | The functions which denote the portion of busy or free resources respectively of the change in the total resource amount at time $t \in \mathbb{R}$, where $\upsilon(t), \mu(t) \in [0,1]$, $\upsilon(t) + \mu(t) = 1.0$. |
| $\gamma(t)$ | The amount of newly requested resources per time unit at time $t \in \mathbb{R}$, where $\gamma(t) > 0$, $\gamma(t) \in \mathbb{R}$. |
| $T_{dyn}$ | A period of a daily fluctuation of the newly requested resources, where $T_{dyn} > 0$. |
| $deg_1$, $deg_2$ | The degrees which determine the level of distinction between the peaks of $\gamma(t)$ over days and weeks, where $deg_1 \in [0,1]$, $deg_2 \in [0,1]$. |
| $T_{pref}$ | The preferable level of temperature in a room, where $T_{pref} \in \mathbb{R}$. |
| $T_{nonst_i}(t)$ | The deviated temperature from $T_{pref}$ at time $t \in \mathbb{R}$ in the case of thermal processes for task $i \in \mathbb{N}$, where $T_{nonst_i}(t) \in \mathbb{R}$. |
| $f'_{dec_i}(t)$, $f'_{inc_i}(t)$ | The damping functions which model the non-stationary decreases and increases of the temperature, compared to the preferable one with the inflection points $t'_{dec_i}(t)$ and $t'_{inc_i}(t)$ for task $i \in \mathbb{N}$, where $f'_{dec_i}(t)$, $f'_{inc_i}(t)$, $t \in \mathbb{R}$. |
| $\tau_i^{dc}(t)$, $\tau_i^{ic}(t)$ | The functions of time which determine a speed of increases and decreases of the temperature for task $i \in \mathbb{N}$, where $\tau_i^{dc}(t)$, $\tau_i^{ic}(t) > 0$, $t \in \mathbb{R}$. |
| $T_{bor_i}(t)$ | The level of temperature up to which the temperature increases or decreases from the preferable one in the case of thermal processes for task $i \in \mathbb{N}$, where $T_{bor_i}(t) \in \mathbb{R}$, $t \in \mathbb{R}$. |
| $T_{amp_i}(t)$, $Ph_i$, $T_{per}^{tem}$ | An amplitude, a phase and period of periodic temperature oscillations around $T_{pref}$ or $T_{nonst_i}(t)$ at time $t \in \mathbb{R}$ for task $i \in \mathbb{N}$, where $T_{amp_i}(t), T_{per}^{tem} > 0$, $T_{amp_i}(t), Ph_i, T_{per}^{tem} \in \mathbb{R}$. |

## 6.4 Evaluation Results for Independent Continuous Tasks

In this section, we discuss the evaluation results for our ConTask negotiation strategy (see Chapter 4), compared to our negotiation strategy (see Chapter 3) which has not been specifically designed for continuous tasks, within a realistically modelled Grid environment as described in Section 6.2. Here, we consider that all tasks are independent in respect of their execution. In Section 6.4.1, we discuss the utilities obtained by a client if the shortening algorithm (see Algorithm 4.2) is used to correct the negotiation outcome, i.e. this algorithm allows a client to concede below the best last GRA's (its own) proposal in order to start the next interruption period at the approximate maximum of resource availability. In Section 6.4.2, we discuss the client utilities when a client uses an evaluation function (see Equation (4.29)) to decide when to concede significantly towards the GRA during negotiation in order to avoid resource exhaustion or prolonged interruptions. Finally, Section 6.4.3 demonstrates the results when both the shortening algorithm and evaluation function are used in negotiation with the GRA. All experiments are evaluated in respect of the standard deviation ($K_3$ in Equation (4.5)) of the GRA's reservation value, which determines the less or more predictable behaviour of the GRA. That is, the larger this standard deviation in respect of the client's maximum value, the less predictable is the GRA's behaviour for a client. The configuration of the client utility has the same parameters as in Chapter 4, and all results are averaged for 100 tasks, where each task is run for one virtual year. Every single negotiation has at most 100 rounds (as discussed in Chapter 3), but it can be repeated until an acceptable duration of allocation period is obtained.

### 6.4.1 Shortening Algorithm

This section compares the client utilities for different deviations of the GRA's reservation value for the cases when a client uses the shortening algorithm "wShort" and when it does not use this algorithm "noShort" as depicted in Figure 6.6. The case "noShort" demonstrates the results when a client applies the negotiation strategy described in Chapter 3, where a client only takes into account a risk of resource exhaustion during negotiation based on the tendencies in resource dynamics. The case "wShort" additionally allows a client to start the next interruption period at the approximate maximum of resource availability. As this figure shows the shortening algorithm improves client utility for all GRA reservation value deviations. However, the difference

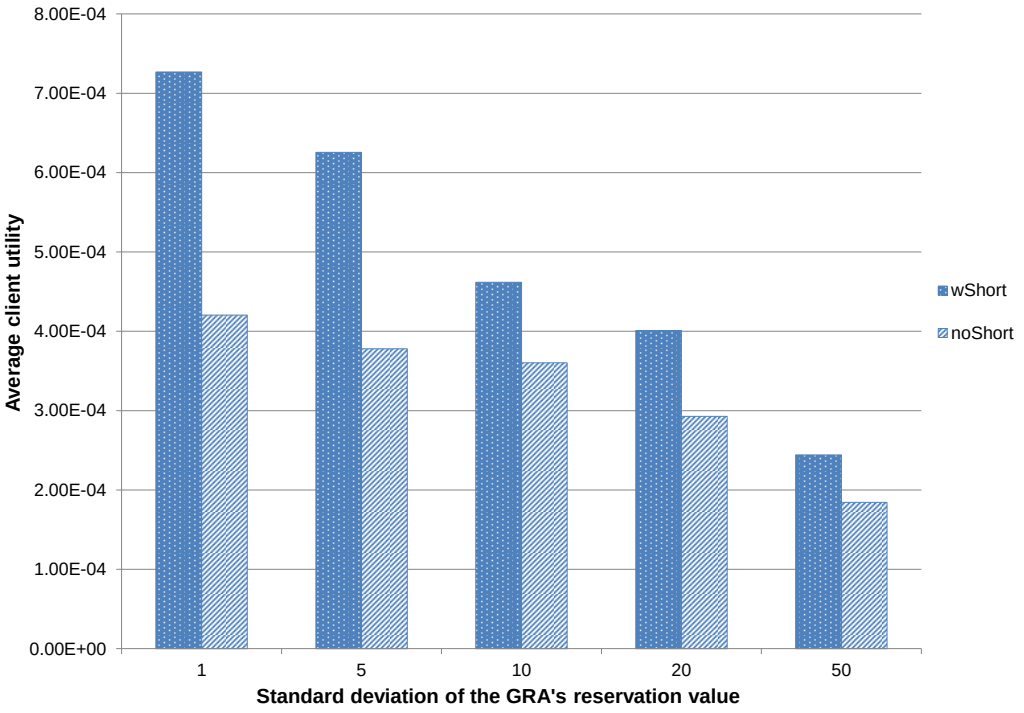
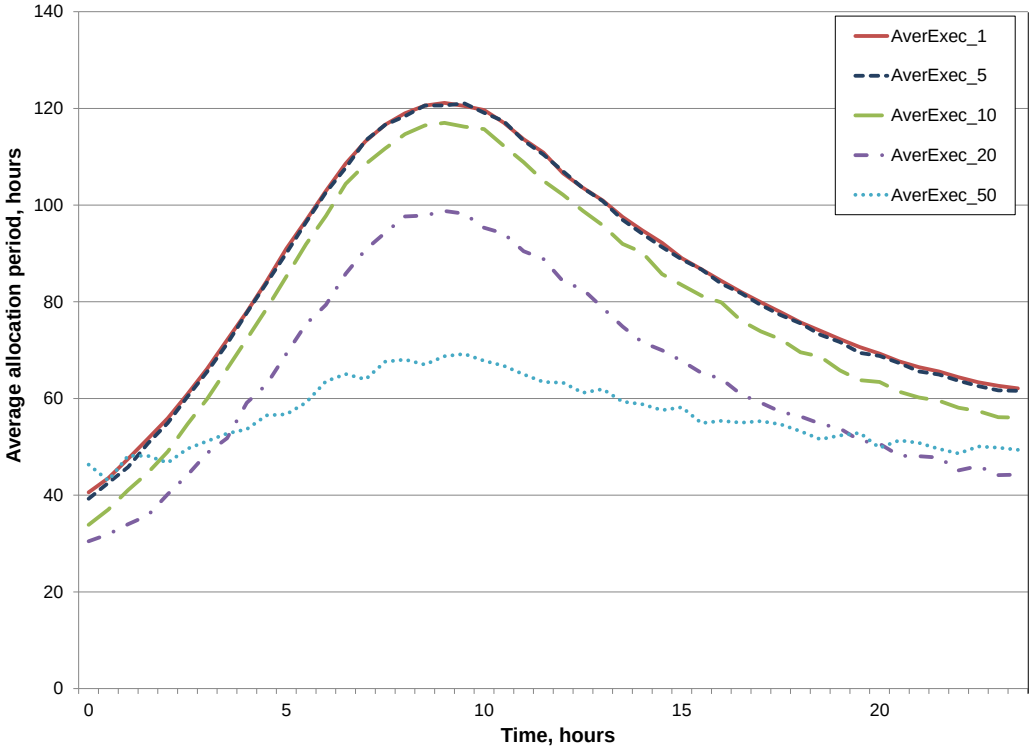


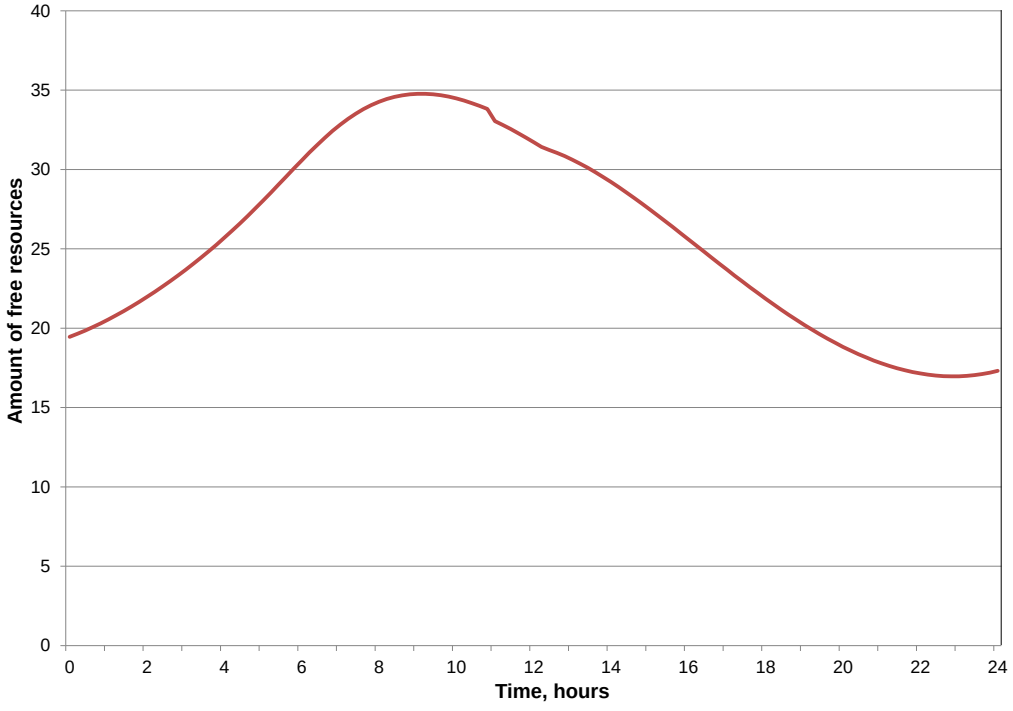

FIGURE 6.6: Evaluation of the shortening algorithm (see Algorithm 4.2)

between utilities in "wShort" and "noShort" becomes gradually smaller with larger intervals of deviations, due to the increasing uncertainty in estimations of maximum resource availability when the GRA's reservation value has significant oscillations. All utilities decrease towards the higher deviation intervals, because the higher level of deviation of the GRA's reservation value denotes that the resources may be exhausted during one or few negotiation rounds when a client has no time to adapt to the resource availability changes.

Figure 6.7(a) shows the allocation periods ("AverAll") averaged over 100 tasks per half an hour for one day for different deviation intervals of the GRA's reservation value in respect of the client's maximum value, where the standard deviation is chosen to be 1% (i.e. "AverAll_1"), 5%, 10%, 20% and 50%. These deviations have been chosen in order to show a trend in the client utility from less to more dynamic Grid environments. The larger deviations mean less determinism in the resource availability periodicity which diminishes the usage of our strategy as demonstrated in our evaluation. Smaller deviations (e.g. "AverAll_1" and "AverAll_5") lead to more accurate estimation of




(a) The change of average allocation periods during one day




(b) The change of resource availability during one day

FIGURE 6.7: An approximation of maximum resource availability

maximum resource availability based on the average allocation periods as depicted in Figure 6.7(a) compared to the maximum resource availability depicted in Figure 6.7(b) during one day. Moreover, it also means that longer allocation periods can be obtained at the maximum of resource availability for the smaller deviations as opposite to the cases with larger deviations. However, the longest allocation periods around the minimum of resource availability are shown for the largest deviation (i.e. "AverAll_50"), which means that larger deviations of the GRA's reservation value can be positive for a client when resource availability is scarce. For example, the GRA may decide to allocate all available resources to a client, causing a large sudden increase in its reservation value. If the deviations are small, this means that the GRA intends not to change its reservation value drastically and as a result it may not offer an acceptable amount of resources to a client when the resources are significantly scarce (e.g. availability is around 0) for a longer time.

### 6.4.2 Evaluation Function

This section demonstrates the client utilities for different deviations of the GRA's reservation value when a client uses its extended evaluation function (see Formula (4.29)) in negotiation with the GRA ("wEval"), which considers the durations of single and total interruptions as depicted in Figure 6.8. If an interruption is too long, a client becomes more generous in negotiation in order to reach an agreement faster. Hence, a client determines a sensitivity threshold $\chi^{int} = \chi^{tot}$ (see Section 4.25) in respect of the decrease in client utility after which a client becomes generous. If this threshold is around 0, then a client becomes generous almost from the beginning of interruption, when the utility starts decreasing. If it is around 0.5, then a client becomes generous only when its expected increment in utility for the following allocation period decreases in around two times. If it is around 1, then we assume that it is too late for a client to become generous, because its utility has already been significantly decreased (i.e. an utility is asymptotically close to 0). Therefore, we consider sensitivity threshold 0.5 as an example of extreme greediness for a client ("noShort&wEval_0.5"), while sensitivity 0.1 is considered as an example of large generosity for a client ("noShort&wEval_0.1"). We do not take into account the thresholds larger than 0.5, because they are non-beneficial for a client in terms of the loss in utility. In our evaluation, we also consider the mid-generous sensitivity threshold 0.2 ("noShort&wEval_0.2"), where sensitivity 0.2 means that a client becomes generous much slower than in the case with sensitivity 0.1.

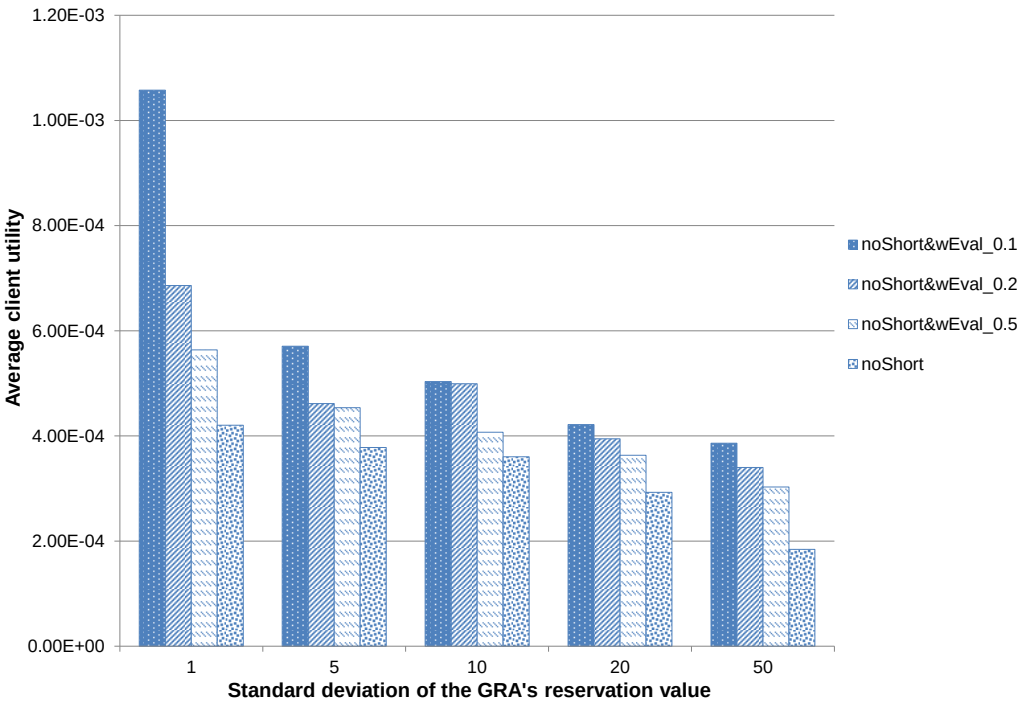



FIGURE 6.8: Client utilities for the extended evaluation function (see Formula (4.29))

Figure 6.8 shows that the smallest utility in almost all cases is when a client uses our previous non-extended evaluation function "noShort", which does not consider the durations of single and total interruptions. This figure also demonstrates that the smaller sensitivity thresholds e.g., 0.1, lead to the higher utilities than the larger thresholds. In the cases of the larger sensitivity thresholds, client's greediness leads to longer interruptions when a client does not necessarily negotiate at the maximum resource availability and, as a result, the smaller utilities. It has to be noted that the difference between utilities for all cases decreases towards the larger intervals of deviation. This can be explained that the large deviations may as much increase the interruptions at any resource availability as decrease it, because of drastic random increases or decreases in the GRA's reservation value. The durations of interruptions on average for the larger deviations are longer , as depicted in Figure 6.9 (e.g. for the cases with standard deviation 5% ("AvrInt_5") and 50% ("AvrInt_50")). However, some long interruptions (e.g. for a couple of hours), caused by resource scarcity, might be smaller due to the possible drastic increase in the GRA's reservation value. In this

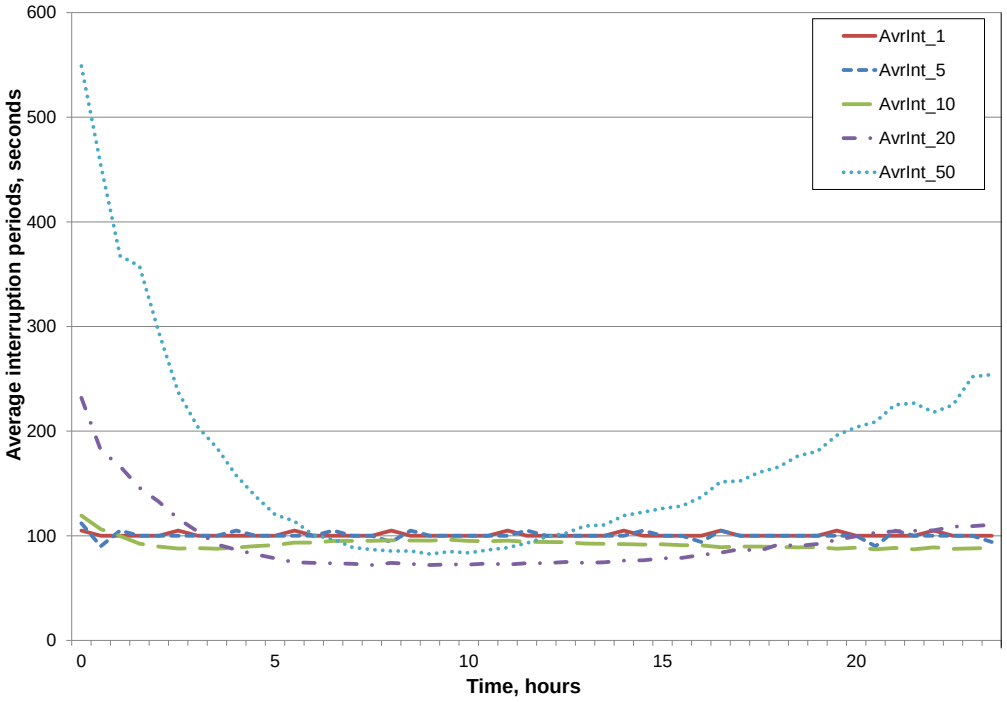



FIGURE 6.9: Average interruption periods for one day

way, the larger deviations of the GRA's reservation value mean more random, and as a result less predictable behaviour of the GRA. Consequently, the client's choice of strategy affects its outcome less than in the cases when the GRA's behaviour is more predictable.

### 6.4.3 ConTask Negotiation Strategy

In this section, we discuss the utilities a client obtains if it uses the shortening algorithm and an extended evaluation function at the same time ("wShort&wEval"), and those utilities are depicted in Figure 6.10. We also compare all cases to the case "noShort", and the figure shows that the ConTask negotiation strategy increases the client utility for all sensitivity thresholds, compared to the case "noShort". It also has to be noted that the utility for the sensitivity threshold 0.1 decreases towards the larger GRA's reservation value deviations, but then it increases again when the standard deviation is equal to 50%. This can be explained as the extremely small deviations e.g., 1%,

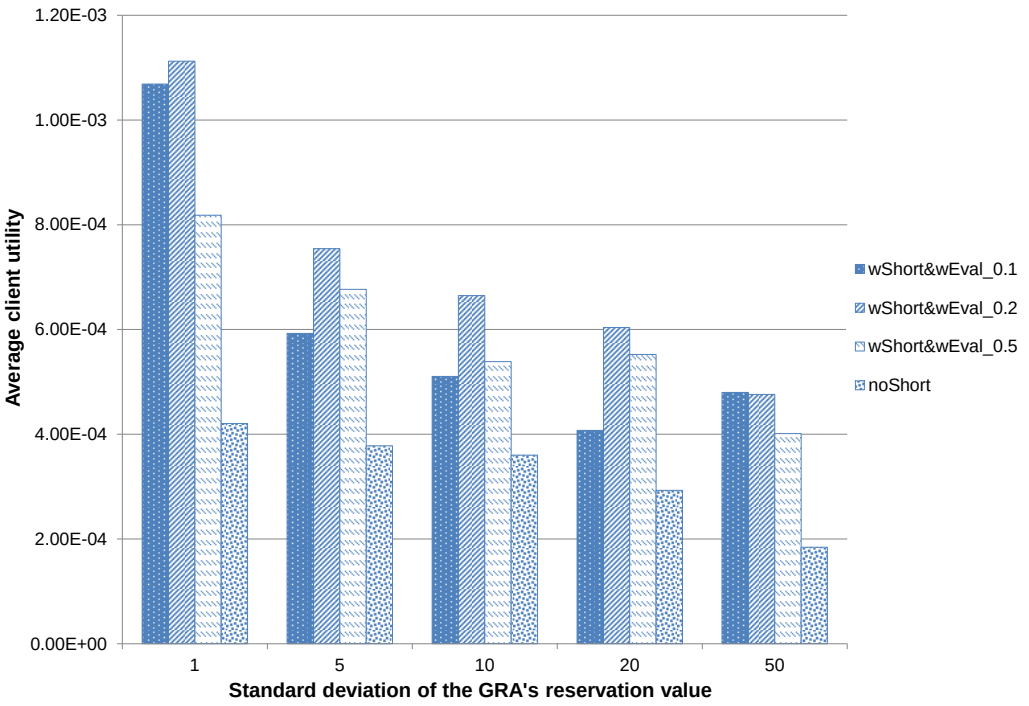



FIGURE 6.10: Evaluation of ConTask negotiation strategy

mean that even the large generosity does not decrease client utility as much as for the larger deviations, because the GRA's reservation value for the small deviations will be always around its maximum in the case when the shortening algorithm is used. On the other hand, the extremely large deviations e.g., 50%, lead to the larger number of negotiation failures and the larger generosity will not decrease the client utility as much as for moderate deviations.

It is observed in Figure 6.10 that all utilities are larger than for the cases when a client uses either the shortening algorithm (see Figure 6.6) or the extended evaluation function (see Figure 6.8).In this way, we can conclude that our ConTask negotiation strategy provides a client with the better utility than its separate components. It is also important to indicate that the sensitivity threshold 0.2 shows the best utilities for almost all considered deviations of the GRA's reservation value. The sensitivity 0.5 leads to the better utilities than 0.1, when the standard deviations are 5%, 10% and 20%, but it shows the worse utilities than 0.1, when the standard deviations are 1% and 50%. It means that an extreme greediness is better than a substantial generosity

for almost all deviations, except for the extremely small or large deviations, but it is worse than a moderate generosity 0.2. In the case of extremely small deviations a greedy client loses in utility in terms of duration of negotiation for no larger outcome, and in the case of extremely large deviations a greedy client loses in utility, because it is difficult to reach an agreement. A conclusion is that a substantial generosity might be more beneficial for the extreme small or large deviations of the GRA's reservation value, compared to the extreme greediness, while a moderate generosity is better for all cases in general. However, an optimal generosity cannot be chosen precisely and it has to be derived through experimentations for the different Grid environment settings, which can be our further direction of research.

## 6.5 Evaluation Results for Inter-dependent Continuous Tasks

In this section, we discuss an evaluation of the SimTask re-allocation strategy (see Chapter 5) in combination with the ConTask negotiation strategy, compared to the case when our negotiation strategy (see Chapter 3) which has not been designed for continuous tasks is used. In this way, we aim to evaluate the overall contribution of all our strategies (negotiation and re-allocation) together within a realistic Grid environment as described in Section 6.2. In all cases, the tasks are inter-dependent and presented as a tree, which is opposite to our evaluation in Section 6.4, where tasks are considered to be independent. Here, the lowest layer tasks monitor the temperature level which is simulated as described in Section 6.3, while the higher layer tasks aggregate those temperatures as a linear combination of all temperature levels, received from their corresponding lower layer sender-tasks. In the following sections, we discuss the client utilities, when the resource amounts are more (see Section 6.5.1) or less (see Section 6.5.2) scarce during a virtual year. In the first section, the total amount of resources can decrease up to 60 resource units, while in the latter section it can decrease only up to 65 resource units, which means the different minimal values for $N_{tot_1}(t)$ in Equation (6.1). A maximal limit $N_{tot_0}$ of the total resource amount $N_{tot}(t)$ is considered to be equal to 100 resource units.

We also evaluate the resulting utilities for different probabilities of an unexpected interruption, because a large number of unexpected interruptions leads to more tasks requiring resources at the same time. In other words, the interruptions occur more

often and they are generally longer, because of increasingly scarce resources. The SimTask re-allocation strategy is assumed to be used mostly in the cases when a client needs to reach an agreement with the GRA urgently, but it is unable to do it because of resource scarcity. The probabilities of the task's unexpected interruption due to resource failure are chosen such as 1.E-05, 1.E-04 and 1.E-03, where the smaller probability denotes a more stable Grid environment and the larger probability denotes the less reliable Grid environment. Our previous experiments in Chapter 5 also show that a probability smaller than 1.E-05 will lead to the same client utility, because it has approximately the same number of unexpected interruptions and, therefore, the smaller probabilities are not considered. A larger probability than 1.E-03 leads to a Grid environment so unstable that it means almost constant task interruption, which is unrealistic and, therefore, it is not considered.

The re-allocation strategy stops one task and re-allocates this task's resources to another task, which is in urgent need of resource due to prolonged interruption, through negotiation with the GRA. The choice of the task-donor is based on the evaluation of its remaining allocation period (e.g. the average period between the longest available and the shortest acceptable) and its connections in a tree of tasks (e.g. the upper layer task will affect the larger number of tasks if it is stopped than the lower layer task). In our simulation, we consider 40 tasks which are connected hierarchically in 4 layers. For our experiment, we need at least 3 layers in the tree to distinguish between the root-task, intermediate tasks and leaf-tasks. However, we choose 4 layers to have more choices of re-allocation among the tasks in order to demonstrate a full potential of the SimTask re-allocation strategy. The values of two weights $W_1$, $W_2$ (see Section 5.3.2.3) which denote the level of importance of the remaining allocation period ($W_1$) or of a location of the donor-task candidate in the tree ($W_2$) are chosen according to our previous evaluation in Chapter 5, i.e. $W_1 = 0.3$ and $W_2 = 0.7$ as this combination has shown the best result in the majority of cases.

In this section, we use the best evaluated configuration parameters for a ConTask negotiation strategy, which we have received in the previous section, in order to evaluate an extent of contribution of the SimTask re-allocation strategy, i.e. the sensitivity threshold is chosen to be 0.2, according to Section 6.4.3. We also assume that the maximum resource availability can be estimated almost precisely. In other words, the reason to use the SimTask re-allocation strategy in our evaluation is due to the unexpected interruptions, as a task can be interrupted when resources are scarce. Our evaluation,

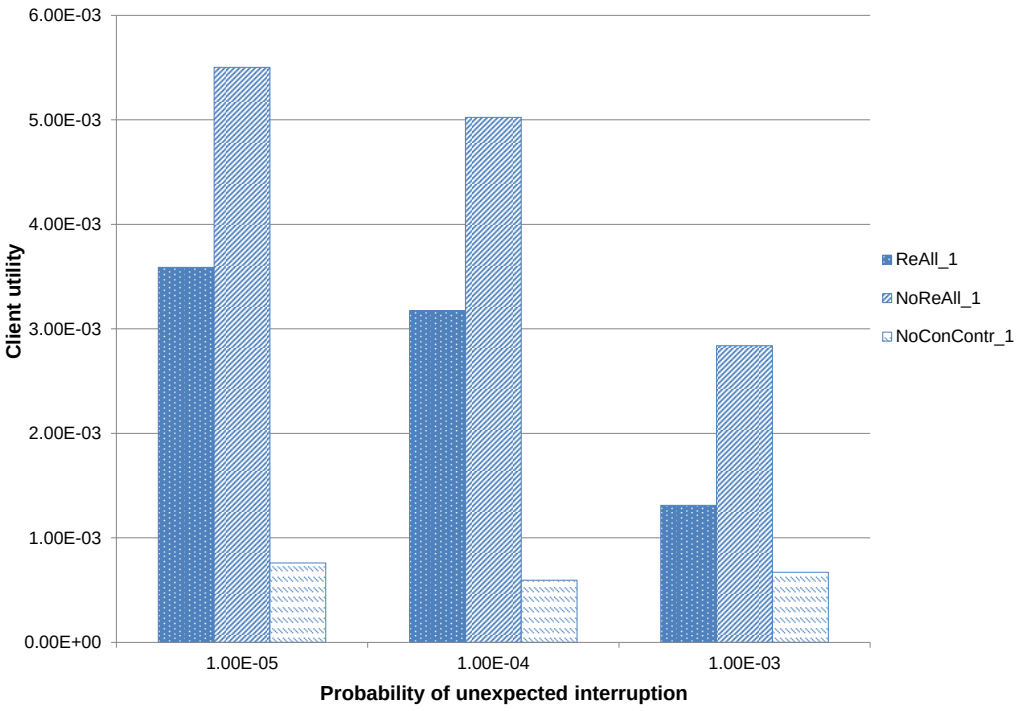



FIGURE 6.11: Client utility for a small deviation of $G_{i,l}^{max}(t)$ in the case of high resource scarcity

as in previous section, also takes into account different deviations of the GRA's reservation value $G_{i,j,l}^{max}(t)$ from the client's maximum value $\tau_{i,l}^{max}$ multiplied on a periodic component $Per(t)$ (see Equation(6.12)), which reflects the level of resource availability at time $t$. The standard deviations are chosen to be 1%, 10% or 50%. A standard deviation 1% denotes a small deviation of the GRA's reservation value, while a standard deviation 50% denotes a large deviation. This large deviation can be occasionally beneficial for a client because it may significantly rise the GRA's reservation value, but in most cases it means highly unpredictable behaviour of the Grid environment. Finally, a standard deviation 10% is chosen as some medium deviation between 1% and 50%, where the impact of deviations is as insignificant as in the case 1% or unpredictable as in the case 50%.

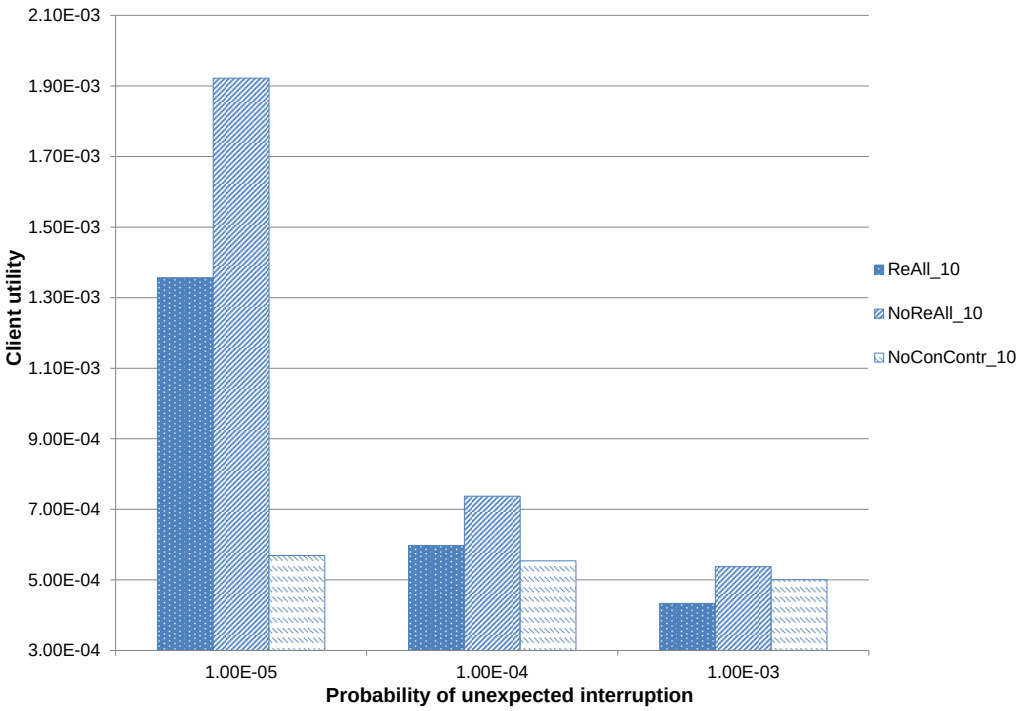



FIGURE 6.12: Client utility for a medium deviation of $G_{i,l}^{max}(t)$ in the case of high resource scarcity

### 6.5.1 High Resource Scarcity

In this section, the client utilities are compared for the cases when the GRA's behaviour is more or less favourable for a client as well as the stability of a Grid environment. The GRA's behaviour is more favourable when its reservation value does not deviate significantly from the client's maximum value, and the Grid environment is more stable when the probability of unexpected interruptions is smaller. In this experiment, a significant resource scarcity (i.e. the resource availability is close to zero) can be observed for the longer durations and it is more frequent than in the following section. In this way, a client generally has more unfavourable conditions for negotiation than in the next experiment even if the probability of unexpected interruptions is small. Figure 6.11 demonstrates client utilities for the case when the GRA's behaviour is more favourable for negotiation in terms of its reservation value deviations, where a standard deviation is equal to 1%. This figure, as well as other figures in this section, compares the cases

when a client uses the SimTask re-allocation strategy together with the ConTask negotiation strategy ("ReAll"), when a client uses only the ConTask negotiation strategy ("NoReAll") and when a client uses only our negotiation strategy which is not designed specifically for continuous or inter-dependent tasks ("NoConContr"), described in Chapter 3. In Figures 6.11 and 6.12, the cases "NoReAll_1" and "NoReAll_10" show the best utilities for all probabilities of an unexpected interruption, where Figure 6.12 depicts the cases when the GRA's behaviour is neither too favourable nor too unfavourable for negotiation with a standard deviation equal to 10%. Nevertheless, Figure 6.13, which demonstrates the client utilities when the GRA's behaviour is highly unfavourable with a standard deviation 50% of its reservation value, shows that the SimTask re-allocation strategy "ReAll_50" improves the utility compared to the cases "NoReAll_50" and "NoConContr_50" for almost all probabilities of an unexpected interruption.

On the one hand, these results support an assumption that it is necessary to use the SimTask re-allocation strategy only when it is almost impossible to reach an agreement with the GRA through ordinary negotiation, i.e. it is difficult to predict the GRA's behaviour. On the other hand, an improvement in the client's utility in the case "ReAll" for large deviations of the GRA's reservation value also can be explained by its possible large increases, which eventually decreases the duration of interruption in these settings. We have argued before that the large deviations of the GRA's reservation value may not lead only to the larger probability of negotiation failure due to its large unexpected decreases, but they may also lead to a negotiation success due to its large increases. Although the large deviations' negative impact on negotiation with a client is more substantial, their positive impact can be seen for substantially long periods of resource scarcity (e.g. days), when the GRA may decide to allocate all remaining resources to this particular client. It also has to be noted that the ConTask negotiation strategy shows an improvement in the client utility even for the large probabilities of unexpected interruption, compared to "ReAll" and "NoConContr", for the majority of cases.

### 6.5.2 Low Resource Scarcity

In this section, we compare client utilities where there is generally less scarce resources in a Grid than in the previous section. In this way, there are less periods of time when resources are substantially scarce and those periods are also shorter. Figures 6.14,

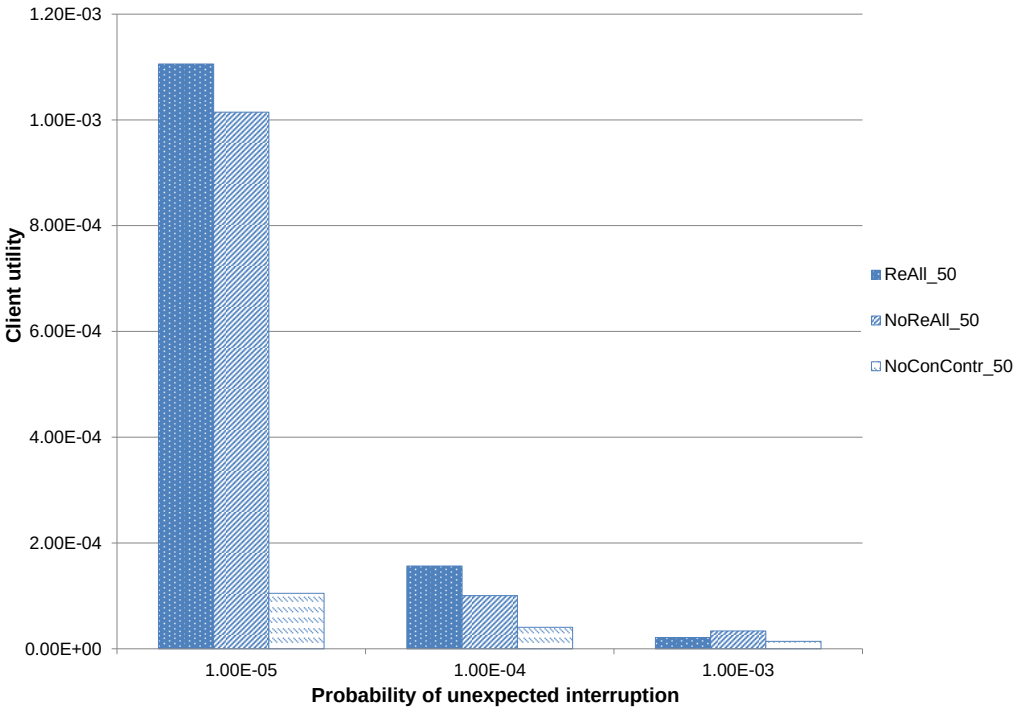



FIGURE 6.13: Client utility for a large deviation of $G_{i,l}^{max}(t)$ in the case of high resource scarcity

6.15 and 6.16 show client utilities for cases when the standard deviation of the GRA's reservation value is 1%, 10% and 50% from the client's maximum value. Here, the cases "ReAll" when the SimTask re-allocation strategy is applied shows better utilities for the smallest probability of an unexpected interruption in the cases "ReAll_10" and "ReAll_50". Generally, the differences between "ReAll" and "NoReAll" are smaller than these differences in the case of high resource scarcity discussed in the previous section. However, in all other cases the best utilities are produced when only the ConTask negotiation strategy is applied.

An improvement, shown in the client utilities for the cases "ReAll_10" and "ReAll_50" when a probability of an unexpected interruption is small, can be explained by the larger number of choices for a task-donor than for the cases when this probability is large. There is also the larger probability of reaching an agreement with the GRA

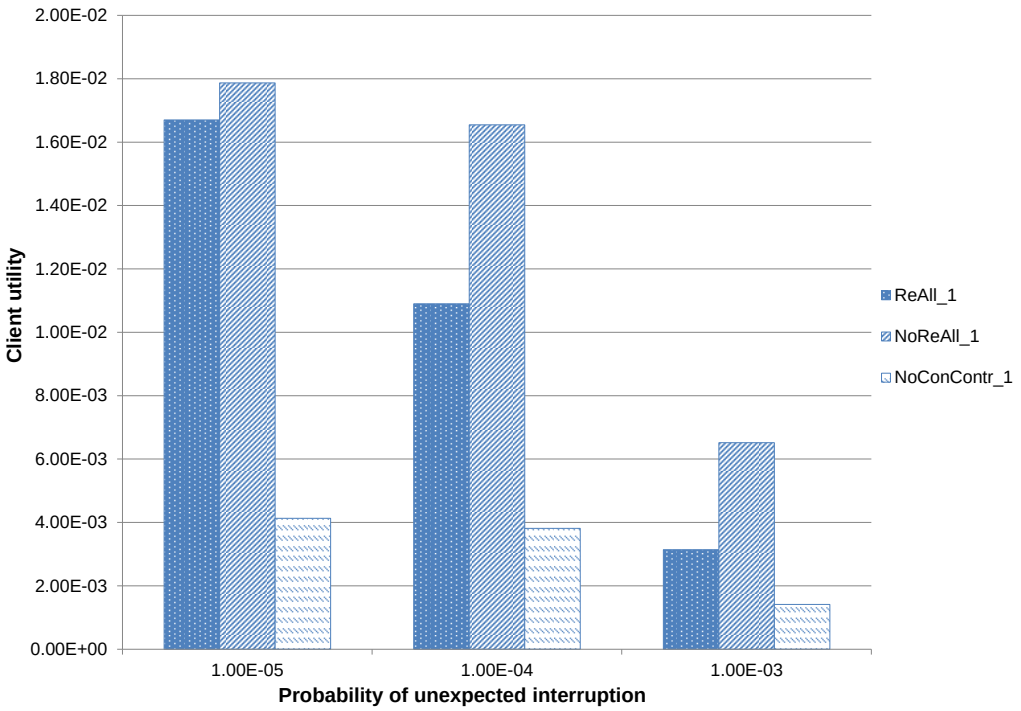



FIGURE 6.14: Client utility for a small deviation of $G_{i,l}^{max}(t)$ in the case of low resource scarcity

for a donor-task than in the previous section which breaks a cycle of resource interchanges among tasks. In other words, this evaluation shows that the resource interchange among tasks is reasonable when there is the larger probability for a resource donating task to obtain resources from the GRA. Otherwise, this donor-task might require resources from another task after some time, etc., which eventually may increase an error of parameter estimation for the upper-layer tasks. The resource re-allocation among client's tasks can be beneficial, when there are more options to choose from in terms of the donor-tasks.

## 6.6 Conclusions

In this chapter, we have described our model of resource dynamism in a Grid, which is based on the observations of realistic periodic patterns in resource availability fluctuations, where a periodicity over days and weeks is considered. A client considers

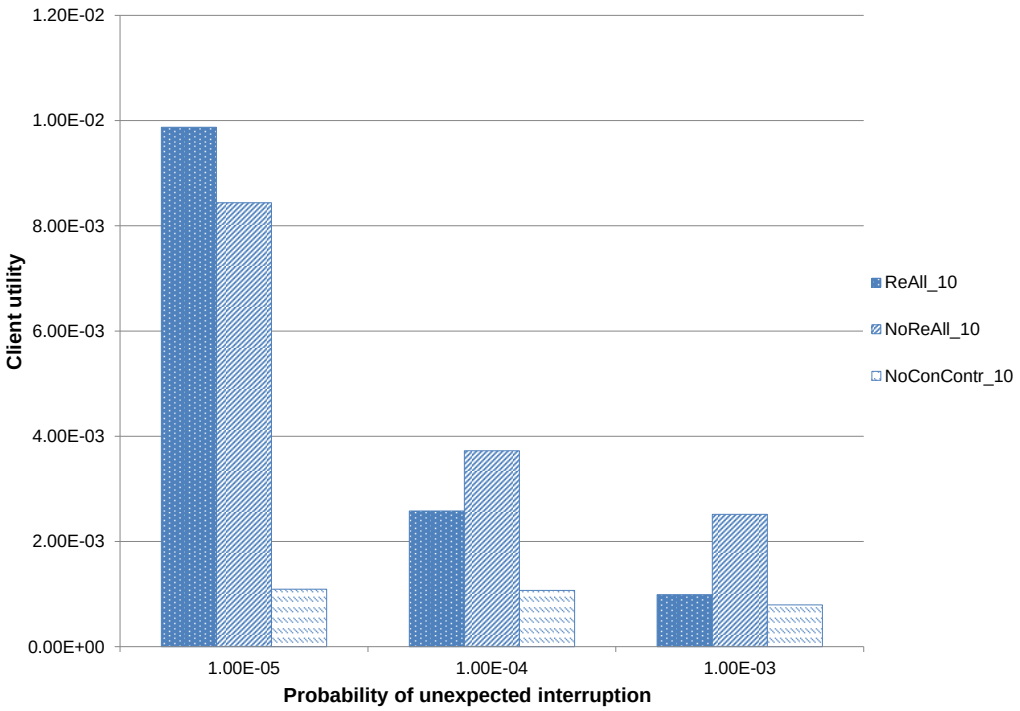



FIGURE 6.15: Client utility for a medium deviation of $G_{i,l}^{max}(t)$ in the case of low resource scarcity

this periodicity in order to predict maximum resource availability in future. We also base our case study on a realistic model of temperature estimation in a building, where temperature fluctuations over time are modelled, considering the real-life dependencies as well as other possible scenarios such as open windows, etc. The temperature modelling has been chosen, because it does not necessarily require expert knowledge and this parameter does not change instantly which allowed us to model acceptable interruptions. The case study evaluated the client utilities in respect of the different Grid environments with larger or smaller resource scarcity over time as well as the different levels of predictability of the GRA's behaviour. We also evaluated our ConTask negotiation strategy and SimTask re-allocation strategy in respect of the different levels of resource reliability in a Grid by considering the larger or smaller number of unexpected interruptions. This range of settings was chosen to test our strategies under both favourable and unfavourable conditions for negotiation.

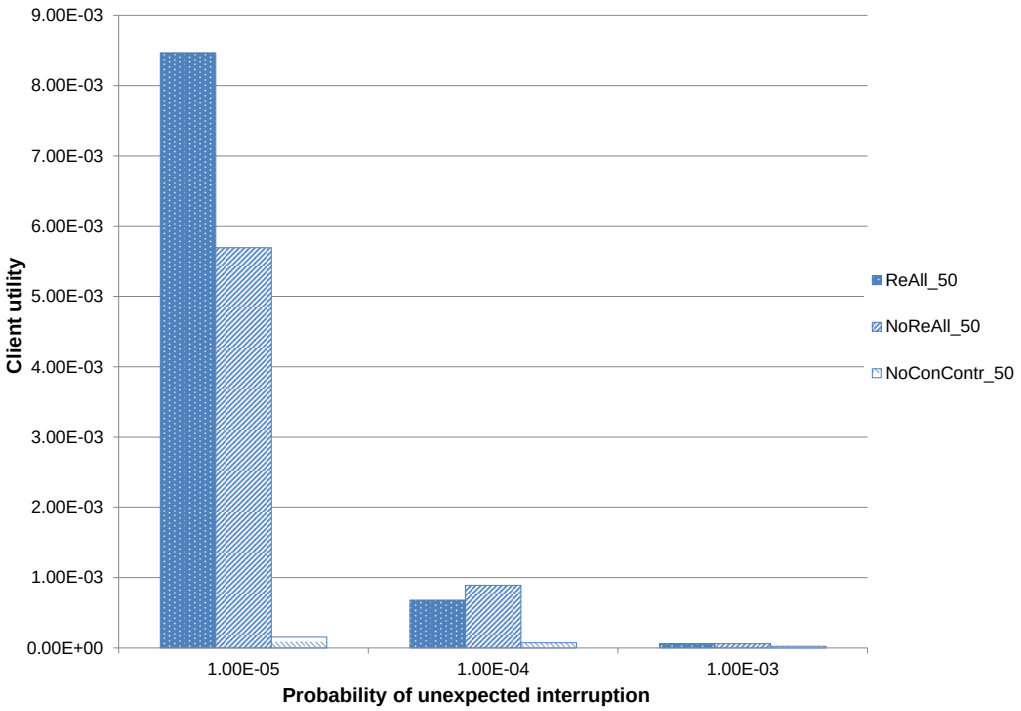



FIGURE 6.16: Client utility for a large deviation of $G_{i,l}^{max}(t)$ in the case of low resource scarcity

The results show that our ConTask negotiation strategy with the considered sensitivity thresholds in respect of the durations of interruption produces better results than our non-continuous task intended negotiation strategy from Chapter 3 for independent or inter-dependent continuous tasks. The ConTask strategy generally shows better utilities than all other strategies evaluated in our case study for the different levels of resource reliability, demonstrating its generic applicability. In particular, it shows more substantial improvements in the client utility for more predictable (smaller deviations) and reliable (smaller number of unexpected interruptions) Grids. The more balanced levels of sensitivity (such as 0.2) show the best results if the maximum resource availability is considered for almost all types of Grids, while substantially generous clients (sensitivity 0.1) benefit more in extremely unpredictable Grids (standard deviation 50%) or when a maximum resource availability is not considered.

Our SimTask re-allocation strategy together with the ConTask negotiation strategy produces better utilities for a client in some cases compared to the cases when only

ConTask negotiation strategy is used, and it almost always produces better utilities when only our non-continuous task negotiation strategy is used. It has to be noted that the SimTask re-allocation strategy can improve the client utility when it has enough choices for a donor-task, and when this donor-task has a reasonable possibility to obtain resources from the GRA to avoid long and frequent exchanges of resources among the tasks. This means that the SimTask re-allocation strategy is more suitable for Grids with a smaller number of unexpected interruptions (probability 1.00E-05) and shorter extreme resource scarcity ranges when the resource availability is close to zero.

# Chapter 7

# Conclusions and Future Work

The number of applications which require continuous or near-continuous execution over long periods of time is steadily increasing now in different areas of science, industry, smart cities, security, etc. This increase prompts a number of issues, due to the execution being resource intensive, real time sensitive and potentially time unbounded. It is desirable to process the tasks which constitute such applications with no or short interruptions for as long as months or years. Pressure on resources also increases when these tasks are inter-dependent in terms of the data exchanged, which is a common issue for many real life scenarios (e.g. monitoring of traffic congestion at a crossroad).

Grid computing appears as a powerful computational tool, which potentially outperforms any single supercomputer, workstation or cluster. A Grid also follows a concept of resource sharing rather than resource leasing [7], which is advantageous for the long-term resource intensive applications in terms of a monetary cost. A Grid is also a large-scale and distributed system which can be advantageous where geographically distributed source data streams [6] are to be processed. However, a Grid has its limitations, such as potential resource scarcity caused by the large number of clients, the uncertainty for a client about resource availability due to its usually limited access to dynamic resource information, the restricted duration of task execution due to the GRA's policies or high resource dynamism, etc. The issue of continuous task execution becomes even more vital when its interruption causes not just a local execution efficiency problem, but affects the performance of the whole client's application as a result of the tasks' data inter-dependencies. Hence, the inter-dependent tasks are desired to

be run simultaneously or near-simultaneously in order to avoid significant data arrival delays among them with short interruption durations being allowed.

The current literature does not adequately address the above mentioned issues, especially for continuous independent or inter-dependent tasks, particularly with regard to the decision-making of clients. A client usually possesses more information about its tasks than it is possible to pass to the GRA. In addition, a client is solely interested in an effective tasks' execution, while the GRA might be more concerned with other issues such as load balancing. This thesis presents several contributions which aim to enhance a client's ability in improving the efficiency of its tasks' execution through negotiation with the GRA, which allows the client to infer some information about resource availability from the GRA's responses. The negotiation process allows both the client and the GRA to find a mutually acceptable agreement, when their objectives are conflicting. This chapter summarises our contributions in Section 7.1 and describes the directions for our future work in Section 7.2.

## 7.1 Research Conclusions

In this section, we summarise our contributions in respect of the research goals described in Chapter 1.

### 7.1.1 Negotiation under Uncertainty and Resource Scarcity

One of the concerns of this thesis has been resource scarcity in Grids, which might lead to resource exhaustion. Consequently, a client's largely greedy behaviour in negotiation might lead to negotiation failure due to resource exhaustion and, as a result, longer interruptions in task execution. On the other hand, a client's largely conceding behaviour might lead to a less desirable amount of resources to be allocated for its tasks (e.g. a shorter execution period), compared to its maximum value (i.e. the most desirable amount of resources). Therefore, a client strives to find a balance between less and more conceding behaviour, which is measured by its level of greediness. In this way, a client has to become more generous in negotiation with the GRA if resources are expected to be exhausted before a nominal negotiation deadline. This decision becomes more complex due to the uncertainty of clients in respect of the GRA's negotiation parameters, i.e. the GRA's level of greediness and reservation value (i.e.

the largest amount of resources the GRA is willing to offer), which potentially reflect the resource availability fluctuations. For example, the GRA becomes more generous when resources are more available and vice verse. This problem is introduced and the adaptive negotiation strategy for a client, which takes into account the risk of resource exhaustion during negotiation, is offered in Chapter 3.

We assume that a client and the GRA follow the same basic time-dependent concession-based strategy [17], but they use different negotiation parameters, which are estimated according to their objectives. Consequently, we propose a negotiation strategy where a client is able to estimate on-line the change in the GRA's negotiation interval accurately (the difference between its reservation and aspiration values), starting from the early negotiation rounds, without any prior knowledge or belief, but only based on this assumption as opposed to other negotiation strategies with an incomplete knowledge about negotiation opponents. A client then uses an on-line shaped fuzzy membership functions (i.e. the bounds of their fuzzy sets) during negotiation, which include this estimate as its input as well as the client's level of greediness, in order to calculate the new client's level of greediness, which will determine its concession tactic towards the GRA.

The way these fuzzy membership functions are shaped every negotiation round depends on the client's *evaluation function*. This allows the client to predict possible resource exhaustion based on the tendencies in resource availability fluctuations, inferred from the GRA's proposals. That is, a client is able to choose the level of greediness, which reduces this risk. This evaluation function's main advantages are an ability to indicate on-line during negotiation the risk of resource exhaustion to a client and a possibility to be expanded easily if complementary components are introduced (discussed in the next section). Hence, an algorithm of choosing the best combination of fuzzy sets (i.e. their *uncertainty intervals*) on-line during negotiation in order to choose the most beneficial tactic for a client has been introduced in Chapter 3.

According to this evaluation function, a client becomes more generous when the resource availability decreases with a substantial speed, which means that resources can be exhausted before the deadline of negotiation. A client becomes greedy in all other cases, aiming to obtain a larger resource amount. It has to be noted that our strategy outperformed other strategies for almost all tendencies of resource fluctuation.

### 7.1.2 Negotiation over Continuous Long-Term Tasks

A core concern of this thesis has been the continuity or near-continuity of task execution, where a task might process data streams, revise a plan continuously, etc. and require a large amount of resources to be allocated for a long or potentially infinite duration. Although the current literature considers continuous task execution (e.g. processing data streams) as described in Chapter 2, it generally lacks the strategic client's involvement in task allocation and execution, which can be expressed through a negotiation process with the GRA. In fact, the existing negotiation strategies largely overlook models with continuous task execution. Hence, we believe that our work is the first which attempts to tackle the problem of continuous long-term task execution with sophisticated client's decision-making. It also has to be noted that the current continuous task models often do not focus on possible interruptions and how these interruptions affect the application's reliability and stability as well as how they can be minimised or avoided.

In order to enable near-continuous task execution for long periods of time, a novel algorithm has been introduced in Chapter 4. This algorithm, which is called a *shortening algorithm*, allows a client to correct its negotiation outcome by conceding beyond its own or the GRA's best proposal in order to start its next negotiation under more favourable condition, i.e. higher resource availability.In this way, a client might obtain a slightly worse resource allocation (i.e. a shorter execution period) in the current negotiation, but it ensures a larger probability of obtaining a better resource allocation (i.e. a longer execution period) in the next negotiation after the currently allocated time slot ends. Higher resource availability also means that a client might not need to repeat negotiation, because it unlikely fails as the GRA is more generous when resources are less scarce. In this way, this algorithm indirectly allows a client to shorten any possible interruptions as each task starts running as soon as an agreement is reached between the client and the GRA.

Our other contribution, which aims to control interruption durations, has been expressed as an extension to the evaluation function mentioned in Section 7.1.1. This extended version considers not only a risk of resource exhaustion, but also whether the current or total interruptions are too long (i.e. the total interruption denotes a total duration of all previous and current interruptions for a continuous task). For example, if they are too long then the client might become more generous in order to reach an

agreement faster. All of the contributions mentioned in this section, have been incorporated into a negotiation strategy for a client, *ConTask*, which is presented and evaluated in Chapter 4, and has been tested under a realistic scenario in Chapter 6. In both chapters, our strategy showed improvements in the client's utility, compared to strategies which do not consider continuous task related issues such as the duration of interruptions and future negotiations.

### 7.1.3 Task Re-allocation for Continuous Inter-dependent Tasks

As we discussed in Chapter 1, continuous task execution becomes more interruption sensitive when it affects the execution of other task(s) in terms of the data exchanges. As opposed to other work, we consider not only the explicit data dependencies, i.e. when a task needs input data from another task in order to produce its results, but also the implicit data dependencies, i.e. when a task can run without another task's input data but its results can be virtually lost (i.e. not processed in real time) due to an interrupted recipient task. In the case of an implicit data dependence, a sender task is considered to be interrupted when its data cannot be processed due to the interruption of a recipient task, but it still holds its resources. Such sender task interruptions are considered in our utility, decreasing this utility as any conventional interruption when resources are lost by a task, and they increase a pressure on a client to run the recipient task which caused this interruption. The time pressure is also faced by a client when a sender task is interrupted and all direct or indirect recipients are affected. In this case, a recipient task can still run for some time, using the last received data from an interrupted sender task, and produce the results with some relative inaccuracy until this inaccuracy becomes substantial enough for a task to stop.

Existing work considers various algorithms for task re-allocation for different reasons such as load balancing or resource failure, as discussed in Chapter 2. However, it lacks such decision-making mechanisms for a client as any resource allocation or re-allocation is usually attributed to the GRA's scheduling algorithms. This work also often assumes that it is always possible to find another resource to migrate a task to. Nevertheless, when a task needs to be re-allocated due to the end of its allocated time slot or resource failure, there might be no available resources at the moment or they can be highly demanded by other clients. In this case, a client can manage its tasks better than the GRA as it can decide which task to stop and which task to run instead of that stopped task based on its knowledge about the task's nature and its own task

priorities. However, a client still has to negotiate its decision with the GRA as only the GRA can actually allocate the resources, and the requested duration of task execution might be shorter as a result of this negotiation. Hence, a client should decide whether it is worth re-allocating its tasks in terms of its utility.

A novel *task re-allocation* strategy, *SimTask*, has been proposed for a client in Chapter 5. First, this strategy decides whether the task's current interruption is critically long and it is necessary to use such measures as to take resources from another task in order to run this one. A critically long time depends on the amount of utility loss to be considered as critical by a client. Second, the strategy chooses which task to stop (i.e. a *donor-task*) in order to allocate its resources to an interrupted task based on several criteria related to this donor-task: its remaining time of resource utilisation; its location in a task tree; its execution status and its influence on the respective recipient-task(s) (e.g. its level of importance for a recipient-task). All those criteria are used to choose a donor-task which has a reasonable remaining time of resource utilisation and whose interruption is least stressful for the whole client's set of tasks. Then, a client has to negotiate its decision with the GRA in order to re-allocate resources from the chosen donor-task to the interrupted task. In the case of multiple interrupted tasks, the chosen task donates its resources to the one which the GRA and the client agreed first. Then, a new donor task is chosen among the remaining tasks which possess resources, etc.

This re-allocation strategy has been evaluated in Chapter 5 and showed an improvement in the client's utility for almost all scenarios compared to the case when task re-allocation is not used by the client. It has also demonstrated an improvement in the client's utility for some cases when it is used together with the ConTask negotiation strategy in Chapter 6 for a specific resource dynamism scenario. This scenario means that a client has a reasonable number of donor candidates to choose from and a reasonable possibility for an interrupted donor-task to reach an agreement with the GRA in respect of the new resources. Otherwise, a frequent task re-allocation may diminish the positive effect caused by the ConTask negotiation strategy.

## 7.2 Future Work

In this section, we discuss possible directions to extend and improve our research in our future work.

### 7.2.1 Negotiation with Meta-Information

In this thesis, a client has a substantially limited knowledge about Grid environment, which is conveyed through a negotiation process with the GRA. A client also focuses mostly on resource availability fluctuations as the main condition for its decisions. However, many other Grid environment characteristics might be relevant for a client (e.g. a demand on resources) and might potentially improve its decisions, but they are difficult or impossible to estimate based just on the GRA's proposals. Therefore, we plan to explore a negotiation process between a client and the GRA, where the GRA can supply some explanatory meta-information on the top of its proposals which can be exploited by a client to its benefit. This information is not necessarily an argument to persuade a client, but it might be just a context for a client to understand the GRA's motivations.

This interesting idea, which is inspired by an argumentation-based approach in negotiation (see Chapter 2), might lead to the number of research issues to explore in respect of a Grid computing. These issues can be:

- an extent of information which is beneficial for the GRA to reveal and how fast this information may change over time;
- the circumstances under which any information can be revealed to a client (e.g. an overloaded Grid);
- the level of improvement any piece of information can bring to the client's utility, if such improvement is observed;
- the computational overheads which might occur due to the more complex decision-making mechanisms for the GRA and a client, etc.

An additional information naturally leads to a more complex decision-making process for a client as the larger number of input parameters can be considered e.g., the number of input membership functions for our fuzzy model may increase and, as a result, the number and complexity of fuzzy rules, which might require an improved mechanism to be dynamically re-designed on-line such as an evaluation function with the larger number of various components.

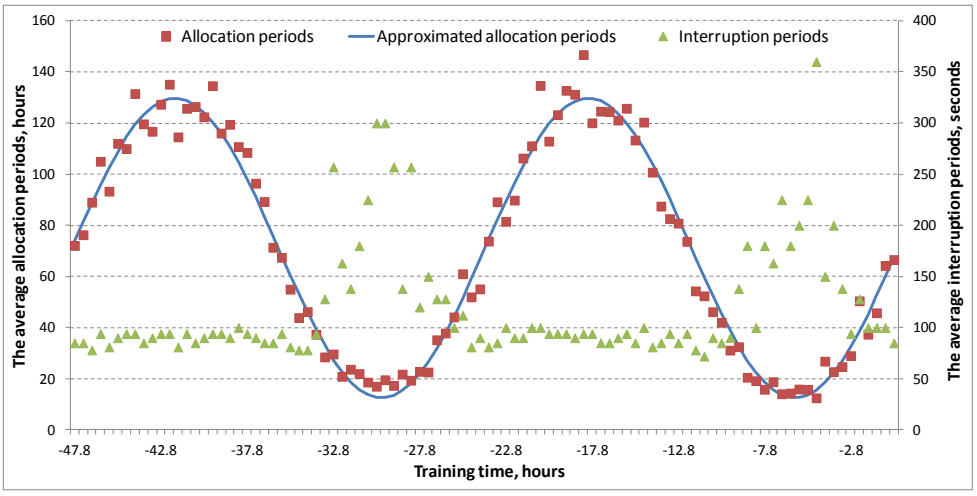



FIGURE 7.1: A demonstration of the periodicity in the interruption periods

### 7.2.2 Negotiation with Identified Unfavourable Time Intervals

In our current work (see Chapter 4), a client always tries to start negotiation from the maximum of resource availability (or close to it) within a continuous task model, but it might be impossible due to, for example, an unexpected resource failure when a client has to negotiate under whatever circumstances its task has been interrupted (e.g. a high level of resource scarcity). Here, the idea is that it might be beneficial for a client to know whether this period of time is *unfavourable* for negotiation right from the start of negotiation in order to avoid prolonged interruptions. For instance, if the risk of resource exhaustion is high, then a client might decide to become generous towards the GRA from the beginning of negotiation. This idea has been supported by our research, which shows specific dependencies in the average interruption durations over resource availability fluctuations as depicted in Figure 7.1.

In Figure 7.1, the average allocation periods denote the agreed with the GRA durations of task execution, while the average interruption periods denote the durations of time which have been spent in order to negotiate these allocation periods. The average allocation periods depict implicitly the level of resource availability at the time of negotiation as the longer periods are allocated when resources are more available. At the same time, the negotiation durations (as a result, interruptions) are longer for the time intervals where resources are less available. Here, the most interesting dependence is that the average interruption periods during the time intervals of less scarce resources

are all around the same duration, which is close to the maximal possible duration of a single negotiation. In this case, a single negotiation is limited to 100 conventional seconds and if a client fails this negotiation, it has to start a new one until it reaches an agreement with the GRA. Therefore, when resources are more available, a client usually needs only one negotiation to reach an agreement, while it may need to repeat negotiations many times for the cases of scarce resources. The implementation of this idea is underway and it is expected to be one of the components of our evaluation function.

### 7.2.3 Re-allocation for Dynamic Task Tree Models

This work (see Chapter 5) considers *static* data inter-dependencies among tasks e.g., a specific lower-layer task has a specific recipient-task, and as a result a static-related re-allocation strategy for a client. For example, our strategy regards an interruption of the upper-layer tasks as more damaging for the client's utility than an interruption of the lower-layer tasks, because such interruption automatically devalues all incoming data from the lower-layer tasks. This happens because the lower-layer tasks cannot choose alternative recipients, but they have only specific ones which can process their data. This situation might change significantly for the recipient-tasks as well, if they can choose alternative sender-tasks in the case when those tasks have been interrupted. We believe that a *dynamic* re-arrangement of task inter-dependencies might lead to the larger robustness and flexibility of task execution, especially, for the continuous long-term tasks.

Consequently, we plan to develop a new model of task inter-dependencies and, as a result, adapt our re-allocation strategy to this model, considering new conditions. For example, our re-allocation strategy might decide when to re-allocate tasks and which task to choose as a donor, taking into consideration possible alternatives for task inter-dependencies. A task re-allocation strategy may not just adapt to the changes in the tasks' inter-dependencies, but it may prompt these changes as an opportunity to avoid task re-allocation, i.e. an interruption of a chosen donor-task.

### 7.2.4 Grid Simulation with Strategic Clients

In our current work (see Chapter 6), a continuous Grid resource simulator is developed which simulates the resource availability, demand on resources and total resource

amount fluctuations over time, based on realistic dependencies observed in the existing literature. However, this simulator does not model multiple clients as autonomous entities with specific strategies, but it models resource demand changes in general without specifying why a particular proposal has been submitted to a Grid and how the GRA has replied to this proposal.

In particular, this improvement might be necessary for modelling multilateral negotiations (e.g. a negotiation strategy for the GRA to negotiate with multiple clients), or some communications / negotiations among clients, while we have focused on a bilateral negotiation in this work. A clients' modelling might also enrich our scenarios in terms of complexity and variability.

# Bibliography


[1] J. F. Sequeda and O. Corcho. Linked stream data: a position paper. In *Proceedings of the 2nd International Workshop on Semantic Sensor Networks*, volume 522, pages 148–157. CEUR-WS, 2009.

[2] R. Motwani, J. Widom, A. Arasu, B. Babcock, S. Babu, M. Datar, G. Manku, C. Olston, J. Rosenstein, and R. Varma. Query processing, resource management, and approximation in a data stream management system. In *Proceedings of the First Biennial Conference on Innovative Data Systems Research*, pages 245–256, 2003.

[3] D. Le-Phuoc, H. Q. Nguyen-Mau, J. X. Parreira, and M. Hauswirth. A middleware framework for scalable management of linked streams. *Web Semantics: Science, Services and Agents on the World Wide Web*, 16(0):42 – 51, 2012. The Semantic Web Challenge 2011.

[4] J. McCormick, N. Belov, P. DiBona, and J. Patti. Plan maintenance for continuous execution management. In *Proceedings of the 15th International Command and Control Research and Technology Symposium*, pages n/a–n/a. Santa Monica, CA, June 2010.

[5] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *International Journal of High Performance Computing Applications*, 15:200–222, 2001.

[6] L. Chen, K. Reddy, and G. Agrawal. GATES: a grid-based middleware for processing distributed data streams. In *Proceedings of the 13th IEEE International Symposium on High Performance Distributed Computing*, pages 192–201, 2004.

[7] M. Armbrust, A. Fox, R. Griffith, Anthony D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. A view of cloud computing. *Communications of the ACM*, 53(4):50–58, April 2010.

[8] J. Gama, P. P. Rodrigues, and L. Lopes. Clustering distributed sensor data streams using local processing and reduced communication. *Intelligent Data Analysis*, 15(1):3–28, 2011.

[9] K. M. Sim. From market-driven e-negotiation to market-driven G-negotiation. In *Proceedings of the 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service*, pages 408–413. IEEE Computer Society, 2005.

[10] N.R. Jennings, P. Faratin, A.R. Lomuscio, S. Parsons, M.J. Wooldridge, and C. Sierra. Automated negotiation: Prospects, methods and challenges. *Group Decision and Negotiation*, 10:199–215, 2001.

[11] A. Iosup, H. Li, M. Jan, S. Anoep, C. Dumitrescu, L. Wolters, and D. H. J. Epema. The grid workloads archive. *The Journal of Future Generation Computer System*, 24(7):672–686, July 2008.

[12] R. Buyya, S. Chapin, and D. DiNucci. Architectural models for resource management in the grid. In Rajkumar Buyya and Mark Baker, editors, *Grid Computing - GRID 2000*, volume 1971 of *Lecture Notes in Computer Science*, pages 18–35. Springer Berlin / Heidelberg, 2000.

[13] H. Li. Workload dynamics on clusters and grids. *The Journal of Supercomputing*, 47(1):1–20, 2009.

[14] C.-B. Cheng, C.-C. H. Chan, and K.-C. Lin. Intelligent agents for e-marketplace: Negotiation with issue trade-offs by fuzzy inference systems. *Decision Support Systems*, 42(2):626–638, November 2006.

[15] S. Akioka and Y. Muraoka. Extended forecast of CPU and network load on computational Grid. In *Proceedings of the 2004 IEEE International Symposium on Cluster Computing and the Grid*, CCGrid 2004, pages 765–772, April 2004.

[16] S. Di, D. Kondo, and W. Cirne. Characterization and comparison of cloud versus grid workloads. In *Proceedings of the 2012 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 230–238, Sept 2012.

[17] P. Faratin, C. Sierra, and N. R. Jennings. Negotiation decision functions for autonomous agents. *Robotics and Autonomous Systems*, 24(3-4):159 –182, 1998.

[18] K. M. Sim. A survey of bargaining models for grid resource allocation. *SIGecom Exchanges*, 5(5):22–32, 2006.

[19] K. M. Sim. Evolving fuzzy rules for relaxed-criteria negotiation. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 38(6):1486–1500, 2008.

[20] I. Al-Anbagi, M. Erol-Kantarci, and H.T. Mouftah. A low latency data transmission scheme for smart grid condition monitoring applications. In *Proceedings of the 2012 IEEE Electrical Power and Energy Conference (EPEC)*, pages 20–25, Oct 2012.

[21] V. Lesser, K. Decker, T. Wagner, N. Carver, A. Garvey, B. Horling, D. Neiman, R. Podorozhny, M.Nagendra Prasad, A. Raja, R. Vincent, P. Xuan, and X.Q. Zhang. Evolution of the GPGP/TAEMS domain-independent coordination framework. *Autonomous Agents and Multi-Agent Systems*, 9(1-2):87–143, 2004.

[22] The Apache Software Foundation. Storm - distributed and fault-tolerant realtime computation. Available from: $\langle http://storm.incubator.apache.org/\rangle$, Visited in June 2014.

[23] A. Litke, D. Skoutas, K. Tserpes, and T. Varvarigou. Efficient task replication and management for adaptive fault tolerance in mobile grid environments. *Future Generation Computer Systems*, 23(2):163 – 178, 2007.

[24] H. Zhao and R. Sakellariou. A low-cost rescheduling policy for dependent tasks on grid computing systems. In *Proceedings of the European Across Grids Conference*, pages 21–31, 2004.

[25] V. Haberland, S. Miles, and M. Luck. Adaptive negotiation for resource intensive tasks in Grids. In K. Kersting and M. Toussaint, editors, *Proceedings of the 6th Starting AI Researchers' Symposium*, volume 241 of *Frontiers in Artificial Intelligence and Applications*, pages 125–136. IOS Press, 2012.

[26] V. Haberland, S. Miles, and M. Luck. Adjustable fuzzy inference for adaptive grid resource negotiation. In K. Fujita, T. Ito, M. Zhang, and V. Robu, editors, *Next Frontier in Agent-based Complex Automated Negotiation*, volume 596 of *Studies of Computational Intelligence*. Springer Japan, 2015.

[27] V. Haberland, S. Miles, and M. Luck. Negotiation to execute continuous long-term tasks. In T. Schaub, G. Friedrich, and B. O'Sullivan, editors, *Proceedings of the 21st European Conference on Artificial Intelligence*, volume 263 of *Frontiers in Artificial Intelligence and Applications*, pages 1019–1020. IOS Press, 2014.

[28] K. M. Sim, Y. Guo, and B. Shi. BLGAN: Bayesian learning and genetic algorithm for supporting negotiation with incomplete information. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 39(1):198–211, 2009.

[29] J. Chen, D. DeWitt, F. Tian, and Y. Wang. NiagaraCQ: a scalable continuous query system for Internet databases. In *Proceedings of the ACM SIGMOD'00 International Conference on Management of Data*, pages 379–390, 2000.

[30] M. Ghanem, Y. Guo, J. Hassard, M. Osmond, and M. Richards. Sensor grids for air pollution monitoring. In *Proceeding of the 3rd UK e-Science All Hands Meeting*, 2004.

[31] S. Madden, M. A. Shah, J. M. Hellerstein, and V. Raman. Continuously adaptive continuous queries over streams. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*, pages 49–60, New York, NY, USA, 2002. ACM.

[32] F. Majeed, M.S. Mahmood, and M. Iqbal. Efficient data streams processing in the real time data warehouse. In *Proceedings of the 2010 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT)*, volume 5, pages 57 –61, july 2010.

[33] W. Clarke and B. Kovatchev. Statistical tools to analyze continuous glucose monitor data. *Diabetes Technology & Therapeutics*, 11(s1):S–45–S–54, June 2009.

[34] P. M. Castro, A. P. Barbosa-Póvoa, H. A. Matos, and A. Q. Novais. Simple continuous-time formulation for short-term scheduling of batch and continuous processes. *Industrial & Engineering Chemistry Research*, 43(1):105–118, 2004.

[35] M. Ahmadi and P. Stone. A multi-robot system for continuous area sweeping tasks. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1724–1729, 2006.

[36] R. Kuntschke, T. Scholl, S. Huber, A. Kemper, A. Reiser, H.M. Adorf, G. Lemson, and W. Voges. Grid-based data stream processing in e-science. In *Proceedings*

*of the 2nd IEEE International Conference on e-Science and Grid Computing*, page 30, December 2006.

[37] Y. Liu, N. N. Vijayakumar, and B. Plale. Stream processing in data-driven computational science. In *Proceeding of the 7th IEEE/ACM International Conference on Grid Computing*, pages 160 –167, 2006.

[38] S. Babu and J. Widom. Continuous queries over data streams. *SIGMOD Record*, 30(3):109–120, September 2001.

[39] D. F. Barbieri, D. Braga, S. Ceri, E. Della Valle, and M. Grossniklaus. C-SPARQL: SPARQL for continuous querying. In *Proceedings of the 18th International Conference on World Wide Web*, WWW '09, pages 1061–1062, New York, NY, USA, 2009. ACM.

[40] H. Lim and S. Babu. Execution and optimization of continuous queries with cyclops. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pages 1069–1072, New York, USA, 2013. ACM.

[41] M. A. Sharaf, P. K. Chrysanthis, A. Labrinidis, and K. Pruhs. Algorithms and metrics for processing multiple heterogeneous continuous queries. *ACM Transactions on Database Systems*, 33(1):5:1–5:44, March 2008.

[42] B. Babcock, M. Datar, and R. Motwani. Load shedding for aggregation queries over data streams. In *Proceedings of the 20th International Conference on Data Engineering*, pages 350–361, March 2004.

[43] M. Cammert, J. Kramer, B. Seeger, and S. Vaupel. An approach to adaptive memory management in data stream systems. In *Proceedings of the 22nd International Conference on Data Engineering*, pages 137–137, April 2006.

[44] M. F. Mokbel and W. G. Aref. Sole: scalable on-line execution of continuous queries on spatio-temporal data streams. *The International Journal on Very Large Data Bases*, 17(5):971–995, 2008.

[45] A. Sallam, K. Nagi, M. Abougabal, and W. Aref. Distributed processing of continuous spatiotemporal queries over road networks. *Alexandria Engineering Journal*, 51(2):85 – 93, 2012.

[46] E. Deelman, S. Callaghan, E. Field, H. Francoeur, R. Graves, N. Gupta, V. Gupta, T.H. Jordan, C. Kesselman, P. Maechling, J. Mehringer, G. Mehta,

D. Okaya, K. Vahi, and L. Zhao. Managing large-scale workflow execution from resource provisioning to provenance tracking: The CyberShake example. In *Proceedings of the 2nd IEEE International Conference on e-Science and Grid Computing*, e-Science '06, pages n/a–n/a, 2006.

[47] P. Neophytou, P. K. Chrysanthis, and A. Labrinidis. Towards continuous workflow enactment systems. In E. Bertino and J. B.D. Joshi, editors, *Collaborative Computing: Networking, Applications and Worksharing*, volume 10 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 162–178. Springer Berlin Heidelberg, 2009.

[48] P. Neophytou, P. K. Chrysanthis, and A. Labrinidis. A continuous workflow scheduling framework. In *Proceedings of the 2nd ACM SIGMOD Workshop on Scalable Workflow Execution Engines and Technologies*, SWEET '13, pages 2:1–2:12, New York, NY, USA, 2013. ACM.

[49] K. L. Myers. Towards a framework for continuous planning and execution. In *Proceedings of the AAAI Fall Symposium on Distributed Continual Planning*, pages n/a–n/a. AAAI Press, 1998.

[50] O. Pettersson. Execution monitoring in robotics: A survey. *Robotics and Autonomous Systems*, 53(2):73 – 88, 2005.

[51] EsperTech Inc. Event series intelligence: Esper & NEsper. Available from: $\langle http://esper.codehaus.org/\rangle$, Visited in June 2014.

[52] TIBCO StreamBase. Available from: $\langle http://www.streambase.com/\rangle$, Visited in June 2014.

[53] D. Le-Phuoc, M. Dao-Tran, J. Xavier Parreira, and M. Hauswirth. A native and adaptive approach for unified processing of linked streams and linked data. In L. Aroyo, C. Welty, H. Alani, J. Taylor, A. Bernstein, L. Kagal, N. Noy, and E. Blomqvist, editors, *The Semantic Web - ISWC 2011*, volume 7031 of *Lecture Notes in Computer Science*, pages 370–388. Springer Berlin Heidelberg, 2011.

[54] K. S. Decker and V. R. Lesser. Generalizing the partial global planning algorithm. *International Journal of Intelligent and Cooperative Information Systems*, 1:319–346, 1992.

[55] V.R. Lesser. A retrospective view of fa/c distributed problem solving. *IEEE Transactions on Systems, Man and Cybernetics*, 21(6):1347 –1362, 1991.

[56] H. Jin, Y. He, W. Wen, and H. Liu. A run-time scheduling policy for dependent tasks in grid computing systems. In *Proceedings of the 6th International Conference on Parallel and Distributed Computing, Applications and Technologies*, pages 521–523, Dec 2005.

[57] L.-T. Lee, C.-W. Chen, H.-Y. Chang, C.-C. Tang, and K.-C. Pan. A non-critical path earliest-finish algorithm for inter-dependent tasks in heterogeneous computing environments. In *Proceedings of the 11th IEEE International High Performance Computing and Communications*, pages 603–608, 2009.

[58] M. Meriem and Y. Belabbas. Dynamic dependent tasks assignment for grid computing. In C.-H. Hsu, L. T. Yang, J. H. Park, and S.-S. Yeo, editors, *Algorithms and Architectures for Parallel Processing*, volume 6082 of *Lecture Notes in Computer Science*, pages 112–120. Springer Berlin Heidelberg, 2010.

[59] F. E. Sandnes and O. Sinnen. Stochastic DFS for multiprocessor scheduling of cyclic taskgraphs. In Kim-Meow Liew, Hong Shen, Simon See, Wentong Cai, Pingzhi Fan, and Susumu Horiguchi, editors, *Parallel and Distributed Computing: Applications and Technologies*, volume 3320 of *Lecture Notes in Computer Science*, pages 354–362. Springer Berlin Heidelberg, 2005.

[60] A. Sardinha, T.A.O. Alves, L.A.J. Marzulo, F.M.G. Franca, V.C. Barbosa, and V.S. Costa. Scheduling cyclic task graphs with scc-map. In *Proceedings of the 3rd Workshop on Applications for Multi-Core Architectures (WAMCA), 2012*, pages 54–59, October 2012.

[61] T. Yang and C. Fu. Heuristic algorithms for scheduling iterative task computations on distributed memory machines. *IEEE Transactions on Parallel and Distributed Systems*, 8(6):608–622, Jun 1997.

[62] S. Franklin and A. Graesser. Is it an agent, or just a program?: A taxonomy for autonomous agents. In Jörg Müller, Michael Wooldridge, and Nicholas Jennings, editors, *Intelligent Agents III Agent Theories, Architectures, and Languages*, volume 1193 of *Lecture Notes in Computer Science*, pages 21–35. Springer Berlin / Heidelberg, 1997.

[63] N. R. Jennings, K. Sycara, and M. Wooldridge. A roadmap of agent research and development. *Autonomous Agents and Multi-Agent Systems*, 1:7–38, 1998.

[64] J. P. Müller. Architectures and applications of intelligent agents: A survey. *The Knowledge Engineering Review*, 13(04):353–380, 1998.

[65] M. Wooldridge and N. R. Jennings. Agent theories, architectures, and languages: A survey. In Michael Wooldridge and Nicholas Jennings, editors, *Intelligent Agents*, volume 890 of *Lecture Notes in Computer Science*, pages 1–39. Springer Berlin / Heidelberg, 1995.

[66] M. Wooldridge and N. R. Jennings. Intelligent agents: theory and practice. *The Knowledge Engineering Review*, 10:115–152, 1995.

[67] R. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14 – 23, March 1986.

[68] M. Sbisa. *Key notions for pragmatics*, chapter "Speech act theory", pages 229 – 244. Handbook of Pragmatics Highlights. John Benjamins Publishing Company, 2009.

[69] J. Allwood. A critical look at speech act theory. In *Logic, Pragmatics and Grammar. Studentlitteratur*, pages 53–69. In Dahl, (Ed.), 1977.

[70] K. Krauter, R. Buyya, and M. Maheswaran. A taxonomy and survey of grid resource management systems for distributed computing. *Software: Practice and Experience*, 32(2):135–164, 2002.

[71] K. Chard, S. Caton, O. Rana, and K. Bubendorfer. Social cloud: Cloud computing in social networks. In *Proceedings of the 2010 IEEE 3rd International Conference on Cloud Computing (CLOUD)*, pages 99–106, July 2010.

[72] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee. Measurement and analysis of online social networks. In *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*, IMC '07, pages 29–42, New York, NY, USA, 2007. ACM.

[73] R. Buyya, Chee S. Y., and S. Venugopal. Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities. In *Proceedings of th 10th IEEE International Conference on High Performance Computing and Communications*, pages 5–13, Sept 2008.

[74] I. Foster, C. Kesselman, J.M. Nick, and S. Tuecke. Grid services for distributed system integration. *Computer*, 35(6):37 –46, June 2002.

[75] T. J. Norman, A. Preece, S. Chalmers, N. R. Jennings, M. Luck, V. D. Dang, T. D. Nguyen, V. Deora, J. Shao, W. A. Gray, and N. J. Fiddian. Agent-based formation of virtual organisations. *Knowledge-Based Systems*, 17(2-4):103–111, 2004. AI 2003, the 23rd SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence.

[76] S.-L. Ding, L.-P. Liu, and J.-B. Yuan. Resource discovery based on multi-agent graph structure in Grid. In *Proceedings of the 2007 International Conference on Machine Learning and Cybernetics*, volume 1, pages 72 –76, Hong Kong, August 2007.

[77] A. Hameurlain, D. Cokuslu, and K. Erciyes. Resource discovery in grid systems: a survey. *International Journal of Metadata, Semantics and Ontologies*, 5(3):251–263, 2010.

[78] G. Kakarontzas and I.K. Savvas. Agent-based resource discovery and selection for dynamic grids. In *Proceedings of the 15th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 195 –200, 2006.

[79] O. F. Rana and L. Moreau. Issues in building agent based computational grids. In *Proceedings of the 3rd Workshop of the UK Special Interest Group on Multi-Agent Systems*, December 2000.

[80] W. Shen, Y. Li, H. H. Ghenniwa, and C. Wang. Adaptive negotiation for agent-based grid computing. In *Proceedings of AAMAS 2002 Workshop on Agentcities: Challenges in Open Agent Environments*, pages 32–36, Bologna, Italy, 2002.

[81] I. Foster, N. R. Jennings, and C. Kesselman. Brain meets brawn: Why grid and agents need each other. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, volume 1 of *AAMAS '04*, pages 8–15, Washington, DC, USA, 2004. IEEE Computer Society.

[82] C. Jonquet, P. Dugenie, and S. Cerri. Service-based integration of grid and multi-agent systems models. In R. Kowalczyk, M. Huhns, M. Klusch, Z. Maamar, and Q. Vo, editors, *Service-Oriented Computing: Agents, Semantics, and Engineering*, volume 5006 of *Lecture Notes in Computer Science*, pages 56–68. Springer Berlin / Heidelberg, 2008.

[83] R. Perrey and M. Lycett. Service-oriented architecture. In *Proceedings of the IEEE/IPSJ International Symposium on Applications and the Internet Workshops*, pages 116 – 119, January 2003.

[84] A. Andrzejak and M. Ceyran. Characterizing and predicting resource demand by periodicity mining. *The Journal of Network and Systems Management*, 13(2):175–196, 2005.

[85] A. Iosup, C. Dumitrescu, D. Epema, H. Li, and L. Wolters. How are real grids used? The analysis of four grid traces and its implications. In *Proceedings of the 7th IEEE/ACM International Conference on Grid Computing*, pages 262–269, Sept 2006.

[86] H. Li. Realistic workload modeling and its performance impacts in large-scale escience grids. *IEEE Transactions on Parallel and Distributed Systems*, 21(4):480–493, 2010.

[87] H. Li and L. Wolters. Towards a better understanding of workload dynamics on data-intensive clusters and grids. In *Proceedings of the 2007 IEEE International Parallel and Distributed Processing Symposium*, pages 1–10, March 2007.

[88] P. A. Dinda. The statistical properties of host load. *The Journal of Scientific Programming*, 7(3-4):211–229, August 1999.

[89] D. Kondo, M. Taufer, C. Brooks, H. Casanova, and A. Chien. Characterizing and evaluating desktop grids: an empirical study. In *Proceedings of the 18th International Parallel and Distributed Processing Symposium*, pages n/a–n/a, 2004.

[90] D. G. Feitelson. Workload modeling for performance evaluation. In *Tutorial Lectures on Performance Evaluation of Complex Systems: Techniques and Tools, Performance 2002*, pages 114–141, London, UK, 2002. Springer-Verlag.

[91] AuverGrid. Available from: $\langle http://www.clermont-universite.fr/\rangle$, Visited in July 2014.

[92] F. Berman, R. Wolski, H. Casanova, W. Cirne, H. Dail, M. Faerman, S. Figueira, J. Hayes, G. Obertelli, J. Schopf, G. Shao, S. Smallen, N. Spring, A. Su, and D. Zagorodnov. Adaptive computing on the grid using AppLeS. *IEEE Transactions on Parallel and Distributed Systems*, 14(4):369 – 382, 2003.

[93] A. B. Downey. Using queue time predictions for processor allocation. In D. G. Feitelson and L. Rudolph, editors, *Job Scheduling Strategies for Parallel Processing*, volume 1291 of *Lecture Notes in Computer Science*, pages 35–57. Springer Berlin Heidelberg, 1997.

[94] S. Kounev, R. Nou, and J. Torres. Autonomic QoS-aware resource management in grid computing using online performance models. In *Proceedings of the 2nd International Conference on Performance Evaluation Methodologies and Tools*, volume 48 of *ValueTools '07*, pages 1–10, 2007.

[95] G. R. Nudd, D. J. Kerbyson, E. Papaefstathiou, S. C. Perry, J. S. Harper, and D. V. Wilcox. PACE–a toolset for the performance prediction of parallel and distributed systems. *High Performance Computing Applications*, 14(3):228–251, 2000.

[96] R. Wolski, N. T. Spring, and J. Hayes. The network weather service: a distributed resource performance forecasting service for metacomputing. *Future Generation Computer Systems*, 15(5-6):757–768, 1999.

[97] R. Wolski. Experiences with predicting resource performance on-line in computational grid settings. *SIGMETRICS Performance Evaluation Review*, 30(4):41–49, 2003.

[98] D.P. Spooner, S.A. Jarvis, J. Cao, S. Saini, and G.R. Nudd. Local grid scheduling techniques using performance prediction. *IEEE Proceedings - Computers and Digital Techniques*, 150(2):87 – 96, 2003.

[99] F. Nadeem, R. Prodan, and T. Fahringer. Characterizing, modeling and predicting dynamic resource availability in a large scale multi-purpose grid. In *Proceedings of the 8th IEEE International Symposium on Cluster Computing and the Grid*, CCGRID '08, pages 348–357, May 2008.

[100] Y. Yuan, Y. Wu, G. Yang, and W. Zheng. Adaptive hybrid model for long term load prediction in computational grid. In *Proceedings of the 8th IEEE International Symposium on Cluster Computing and the Grid*, CCGRID '08, pages 340–347, May 2008.

[101] M. Xie, Z. Yun, Z.. Lei, and G. Allen. Cluster abstraction: Towards uniform resource description and access in multicluster grid. In *Proceedings of the Second*

*International Multi-Symposiums on Computer and Computational Sciences*, IMSCCS '07, pages 220–227, Washington, DC, USA, 2007. IEEE Computer Society.

[102] M.D. de Assunção and R. Buyya. Performance analysis of multiple site resource provisioning: Effects of the precision of availability information. In P. Sadayappan, M. Parashar, R. Badrinath, and V.K. Prasanna, editors, *High Performance Computing - HiPC 2008*, volume 5374 of *Lecture Notes in Computer Science*, pages 157–168. Springer Berlin, Heidelberg, 2008.

[103] S. Chapin, D. Katramatos, J. Karpovich, and A. Grimshaw. The legion resource management system. In Dror Feitelson and Larry Rudolph, editors, *Job Scheduling Strategies for Parallel Processing*, volume 1659 of *Lecture Notes in Computer Science*, pages 162–178. Springer Berlin / Heidelberg, 1999.

[104] J. Cao, D.J. Kerbyson, and G.R. Nudd. Performance evaluation of an agent-based resource management infrastructure for grid computing. In *Proceedings of the 1st IEEE/ACM International Symposium on Cluster Computing and the Grid*, pages 311 –318, 2001.

[105] J. Cao, S. A. Jarvis, S. Saini, D. J. Kerbyson, and G. R. Nudd. ARMS: An agent-based resource management system for grid computing. *Scientific Programming*, 10(2):135–148, January 2002.

[106] Z. Shi, H. Huang, J. Luo, F. Lin, and H. Zhang. Agent-based grid computing. *Applied Mathematical Modelling*, 30(7):629 – 640, 2006. Parallel and Vector Processing in Science and Engineering.

[107] Y.-S. Kee, D. Logothetis, R. Huang, H. Casanova, and A Chien. Efficient resource description and high quality selection for virtual grids. In *Proceedings of the 2005 IEEE International Symposium on Cluster Computing and the Grid*, volume 1, pages 598–606, May 2005.

[108] Y.-S. Kee and C. Kesselman. Grid resource abstraction, virtualization, and provisioning for time-targeted applications. In *Proceedings of the 2008 IEEE International Symposium on Cluster Computing and the Grid*, CCGRID '08, pages 324–331, Washington, DC, USA, 2008. IEEE Computer Society.

[109] S. Adabala, V. Chadha, P. Chawla, R. Figueiredo, J. Fortes, I. Krsul, A. Matsunaga, M. Tsugawa, J. Zhang, M. Zhao, L. Zhu, and X. Zhu. From virtualized

resources to virtual computing grids: The In-VIGO system. *Future Generation Computer Systems*, 21(6):896–909, June 2005.

[110] P. Garbacki and V.K. Naik. Efficient resource virtualization and sharing strategies for heterogeneous grid environments. In *Proceedings of the 10th IFIP/IEEE International Symposium on Integrated Network Management*, pages 40–49, May 2007.

[111] R. Wolski, J. Brevik, J. S. Plank, and T. Bryan. *Grid Resource Allocation and Control Using Computational Economies*, chapter 32, pages 747–771. John Wiley & Sons, Ltd, 2003.

[112] V. Narayanan and N. R. Jennings. An adaptive bilateral negotiation model for e-commerce settings. In *Proceedings of the 7th International IEEE Conference on E-Commerce Technology*, pages 34–39, 2005.

[113] R. Raman, M. Livny, and M. Solomon. Matchmaking: distributed resource management for high throughput computing. In *Proceedings of the 7th International Symposium on High Performance Distributed Computing*, pages 140–146, Jul 1998.

[114] K. Czajkowski, I. T. Foster, N. T. Karonis, C. Kesselman, S. Martin, W. Smith, and S. Tuecke. A resource management architecture for metacomputing systems. In *Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing*, IPPS/SPDP '98, pages 62–82, London, UK, 1998. Springer-Verlag.

[115] M. Baker, R. Buyya, and D. Laforenza. Grids and grid technologies for wide-area distributed computing. *Software: Practice and Experience*, 32(15):1437–1466, 2002.

[116] H. Casanova and J. Dongarra. Applying netsolve's network-enabled server. *IEEE Computational Science Engineering*, 5(3):57–67, Jul 1998.

[117] M. Beck, J. Dongarra, and J.S. Plank. Netsolve/d: a massively parallel grid execution system for scalable data intensive collaboration. In *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium*, pages 8 pp.–, April 2005.

[118] H. Nakada, H. Takagi, S. Matsuoka, U. Nagashima, M. Sato, and S. Sekiguchi. Utilizing the metaserver architecture in the ninf global computing system. In

P. Sloot, M. Bubak, and B. Hertzberger, editors, *High-Performance Computing and Networking*, volume 1401 of *Lecture Notes in Computer Science*, pages 607–616. Springer Berlin, Heidelberg, 1998.

[119] Y. Tanaka, H. Nakada, S. Sekiguchi, T. Suzumura, and S. Matsuoka. Ninf-g: A reference implementation of rpc-based programming middleware for grid computing. *The Journal of Grid Computing*, 1(1):41–51, 2003.

[120] N. Wilkins-Diehr, D. Gannon, G. Klimeck, S. Oster, and S. Pamidighantam. Teragrid science gateways and their impact on science. *Computer*, 41(11):32–41, Nov 2008.

[121] D.A Reed. Grids, the teragrid and beyond. *Computer*, 36(1):62–68, Jan 2003.

[122] C.A. Waldspurger, T. Hogg, B.A. Huberman, J.O. Kephart, and W.S. Stornetta. Spawn: a distributed computational economy. *IEEE Transactions on Software Engineering*, 18(2):103 – 117, February 1992.

[123] M. Stonebraker, R. Devine, M. Kornacker, W. Litwin, A. Pfeffer, A. Sah, and C. Staelin. An economic paradigm for query processing and data migration in mariposa. In *Proceedings of the Third International Conference on Parallel and Distributed Information Systems*, pages 58 – 67, September 1994.

[124] D. Abramson, R. Buyya, and J. Giddy. A computational economy for grid computing and its implementation in the Nimrod-G resource broker. *Future Generation Computer Systems*, 18:1061–1074, October 2002.

[125] R. Buyya, D. Abramson, J. Giddy, and H. Stockinger. Economic models for resource management and scheduling in Grid computing. *Concurrency and Computation: Practice and Experience*, 14(13-15):1507–1542, 2002.

[126] P. Anthony and N. R. Jennings. Developing a bidding agent for multiple heterogeneous auctions. *ACM Transactions on Internet Technology*, 3:185–217, August 2003.

[127] S. S. Fatima, M. Wooldridge, and N. R. Jennings. Sequential auctions for common value objects with budget constrained bidders. *Multiagent and Grid Systems*, 6(5):403–414, January 2010.

[128] T. Sandholm. Limitations of the vickrey auction in computational multiagent systems. In *Proceedings of the Second International Conference on Multiagent Systems*. AAAI Press, 1996.

[129] M. Fasli. *Agent Technology for e-Commerce*, chapter "Combinatorial Auctions", pages 236 –239. John Wiley & Sons, Ltd., The Atrium, Southern Gate, Chichester, West Sussex, England, 2007.

[130] M. Schwind, T. Stockheim, and O. Gujo. Agents' bidding strategies in a combinatorial auction controlled grid environment. In Maria Fasli and Onn Shehory, editors, *Agent-Mediated Electronic Commerce. Automated Negotiation and Strategy Design for Electronic Markets*, volume 4452 of *Lecture Notes in Computer Science*, pages 149–163. Springer Berlin / Heidelberg, 2007.

[131] L. Xing and Z. Lan. A method based on iterative combinatorial auction mechanism for resource allocation in grid multi-agent systems. In *Proceedings of the 2009 International Conference on Intelligent Human-Machine Systems and Cybernetics*, volume 1, pages 36 –39, August 2009.

[132] Y. Kong, M. Zhang, and D. Ye. A negotiation-based method for task allocation with time constraints in open grid environments. *Concurrency and Computation: Practice and Experience*, pages n/a–n/a, 2014.

[133] K.Q. Yan, S.C. Wang, C.P. Chang, and J.S. Lin. A hybrid load balancing policy underlying grid computing environment. *Computer Standards & Interfaces*, 29(2):161 – 173, 2007.

[134] E. Elmroth and J. Tordsson. Grid resource brokering algorithms enabling advance reservations and resource selection based on performance predictions. *Future Generation Computer Systems*, 24(6):585 – 593, 2008.

[135] L. Tomas, B. Caminero, and C. Carrion. Bag of tasks rescheduling within real grid environments: Different approaches. In *Proceedings of the 21st Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, pages 213–217, Feb 2013.

[136] B. Rood and M.J. Lewis. Resource availability prediction for improved grid scheduling. In *Proceedings of the IEEE 4th International Conference on eScience*, eScience '08, pages 711–718, Dec 2008.

[137] J. Zhang and C. Phillips. Job-scheduling via resource availability prediction for volunteer computational grids. *The International Journal of Grid and Utility Computing*, 2:25–32, 2011.

[138] P. Muthuchelvi, G. S. Anandha Mala, and V. Ramachandran. Agent based grid resource discovery with negotiated alternate solution and non-functional requirement preferences. *The Journal of Computer Science*, 5(3):191–198, 2009.

[139] A. Haque, S.M. Alhashmi, and R. Parthiban. Continuous double auction in grid computing: An agent based approach to maximize profit for providers. In *Proceedings of the 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, volume 2, pages 347 – 351, September 2010.

[140] H. Izakian, B.T. Ladani, K. Zamanifar, A. Abraham, and V. Snasel. A continuous double auction method for resource allocation in computational grids. In *Proceedings of the 2009 IEEE Symposium on Computational Intelligence in Scheduling*, pages 29 –35, 2009.

[141] R. Lawley, M. Luck, K. Decker, T. R. Payne, and L. Moreau. Automated negotiation between publishers and consumers of grid notifications. *Parallel Processing Letters*, 13(4):537–548, December 2003.

[142] F. Lopes, M. Wooldridge, and A. Novais. Negotiation among autonomous computational agents: principles, analysis and challenges. *Artificial Intelligence Review*, 29:1–44, 2008.

[143] M. J. Osborne and A. Rubinstein. *Bargaining and Markets*. Academic Press, 1990.

[144] K. Chatterjee and H. Sabourian. Multiperson bargaining and strategic complexity. *Econometrica*, 68(6):1491–1509, 2000.

[145] J. S. Rosenschein and G. Zlotkin. *Rules of Encounter: Designing Conventions for Automated Negotiation Among Computers*. MIT Press, Cambridge, MA, USA, 1994.

[146] H. Sabourian. Bargaining and markets: complexity and the competitive outcome. *Journal of Economic Theory*, 116(2):189 – 228, 2004.

[147] D. G. Pruitt. Strategic choice in negotiation. *American Behavioral Scientist*, 27(2):167–194, November 1983.

[148] F. Lang. Developing dynamic strategies for multi-issue automated contracting in the agent based commercial grid. In *Proceedings of the 2005 IEEE International Symposium on Cluster Computing and the Grid*, volume 1, pages 342 –349, 2005.

[149] B. An, V. Lesser, and K. M. Sim. Strategic agents for multi-resource negotiation. *Autonomous Agents and Multi-Agent Systems*, 23:114–153, 2011.

[150] I. Rahwan, S. D. Ramchurn, N. R. Jennings, P. Mcburney, S. Parsons, and L. Sonenberg. Argumentation-based negotiation. *The Knowledge Engineering Review*, 18(4):343–375, December 2003.

[151] F. Sadri, F. Toni, and P. Torroni. Dialogues for negotiation: Agent varieties and dialogue sequences. In J.-J.Ch. Meyer and M. Tambe, editors, *Intelligent Agents VIII*, volume 2333 of *Lecture Notes in Computer Science*, pages 405–421. Springer Berlin, Heidelberg, 2002.

[152] P. McBurney, R.M. Van Eijk, S. Parsons, and L. Amgoud. A dialogue game protocol for agent purchase negotiations. *Autonomous Agents and Multi-Agent Systems*, 7(3):235–273, 2003.

[153] L. Amgoud and S. Parsons. Agent dialogues with conflicting preferences. In J.-J.Ch. Meyer and M. Tambe, editors, *Intelligent Agents VIII*, volume 2333 of *Lecture Notes in Computer Science*, pages 190–205. Springer Berlin, Heidelberg, 2002.

[154] A. Rubinstein. Perfect equilibrium in a bargaining model. *Econometrica*, 50(1):97–109, 1982.

[155] F. Lopes, N. Mamede, A.Q. Novais, and H. Coelho. A negotiation model for autonomous computational agents: Formal description and empirical evaluation. *The Journal of Intelligent and Fuzzy Systems*, 12(3):195–212, January 2002.

[156] T. Ito, M. Klein, and H. Hattori. A multi-issue negotiation protocol among nonlinear utility agents: A preliminary report. In T. Ito, H. Hattori, M. Zhang, and T. Matsuo, editors, *Rational, Robust, and Secure Negotiations in Multi-Agent Systems*, volume 89 of *Studies in Computational Intelligence*, pages 25–38. Springer Berlin, Heidelberg, 2008.

[157] T. D. Nguyen and N. R. Jennings. A heuristic model of concurrent bi-lateral negotiations in incomplete information settings. In *Proceedings of the International Joint Conferences on Artificial Intelligence*, pages 1467–1469, 2003.

[158] T. D. Nguyen and N. R. Jennings. Coordinating multiple concurrent negotiations. In *Proceedings of the Third International Joint Conference on Autonomous*

*Agents and Multiagent Systems*, volume 3 of *AAMAS '04*, pages 1064–1071, Washington, DC, USA, 2004. IEEE Computer Society.

[159] K. M. Sim and B. Shi. Concurrent negotiation and coordination for grid resource coallocation. *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, 40(3):753 –766, 2010.

[160] M. Klein, P. Faratin, H. Sayama, and Y. Bar-Yam. Protocols for negotiating complex contracts. *IEEE Intelligent Systems*, 18(6):32–38, Nov 2003.

[161] G. C. Silaghi, L. D. Şerban, and C. M. Litan. A time-constrained SLA negotiation strategy in competitive computational grids. *Future Generation Computer Systems*, 28(8):1303–1315, 2012.

[162] V. Narayanan and N. R. Jennings. Learning to negotiate optimally in non-stationary environments. In *Proceedings of the 10th international conference on Cooperative Information Agents*, pages 288–300. Springer, 2006.

[163] K. Hindriks and D. Tykhonov. Opponent modelling in automated multi-issue negotiation using bayesian learning. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '08, pages 331–338, 2008.

[164] C. Hou. Predicting agents tactics in automated negotiation. In *Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology, 2004.*, pages 127 – 133, 2004.

[165] S. Chen and G. Weiss. A novel strategy for efficient negotiation in complex environments. In I. Timm and C. Guttmann, editors, *Multiagent System Technologies*, volume 7598 of *Lecture Notes in Computer Science*, pages 68–82. Springer Berlin, Heidelberg, 2012.

[166] S. Kawaguchi, K. Fujita, and T. Ito. Compromising strategy based on estimated maximum utility for automated negotiation agents competition (ANAC-10). In K. G. Mehrotra, C. K. Mohan, J. C. Oh, P. K. Varshney, and M. Ali, editors, *Modern Approaches in Applied Intelligence*, volume 6704 of *Lecture Notes in Computer Science*, pages 501–510. Springer Berlin Heidelberg, 2011.

[167] C. R. Williams, V. Robu, E. H. Gerding, and N. R. Jennings. Using gaussian processes to optimise concession in complex negotiations against unknown opponents. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, volume 1 of *IJCAI 2011*, pages 432–438. AAAI Press, 2011.

[168] D. Zeng and K. Sycara. Bayesian learning in negotiation. *International Journal of Human-Computer Studies*, 48(1):125 – 141, 1998.

[169] J. Gwak and K. M. Sim. Bayesian learning based negotiation agents for supporting negotiation with incomplete information. *Lecture Notes in Engineering and Computer Science*, 2188(1):163–168, 2011.

[170] L. Buşoniu, R. Babuska, and B. de Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 38(2):156–172, March 2008.

[171] P.A Heeman. Representing the reinforcement learning state in a negotiation dialogue. In *Proceedings of the IEEE Workshop on Automatic Speech Recognition Understanding*, pages 450–455, Nov 2009.

[172] R. P. Prado, S. García-Galán, A. J. Yuste, and J. E. Muñoz Expósito. A fuzzy rule-based meta-scheduler with evolutionary learning for grid computing. *Engineering Applications of Artificial Intelligence*, 23(7):1072–1082, October 2010.

[173] C.C. Lee. Fuzzy logic in control systems: fuzzy logic controller. i. *IEEE Transactions on Systems, Man and Cybernetics*, 20(2):404–418, mar/apr 1990.

[174] M. He, H.-f. Leung, and N. R. Jennings. A fuzzy-logic based bidding strategy for autonomous agents in continuous double auctions. *IEEE Transactions on Knowledge and Data Engineering*, 15(6):1345–1363, 2003.

[175] M. He, A. Rogers, X. Luo, and N. R. Jennings. Designing a successful trading agent for supply chain management. In *Proceedings of the 5th International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS 2006, pages 1159–1166, New York, NY, USA, 2006. ACM.

[176] J. Richter, R. Kowalczyk, and M. Klusch. Multistage fuzzy decision making in bilateral negotiation with finite termination times. In Ann Nicholson and Xiaodong Li, editors, *AI 2009: Advances in Artificial Intelligence*, volume 5866 of *Lecture Notes in Computer Science*, pages 21–30. Springer Berlin / Heidelberg, 2009.

[177] E. H. Mamdani. Application of fuzzy logic to approximate reasoning using linguistic synthesis. *IEEE Transactions on Computers*, C-26(12):1182 –1191, 1977.

[178] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338 – 353, 1965.

[179] T. Takagi and M. Sugeno. Fuzzy identification of systems and its applications to modeling and control. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-15(1):116–132, Jan 1985.

[180] C.-F. Juang, J.-Y. Lin, and C.-T. Lin. Genetic reinforcement learning through symbiotic evolution for fuzzy controller design. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 30(2):290–302, Apr 2000.

[181] M. Sugeno and T. Yasukawa. A fuzzy-logic-based approach to qualitative modeling. *IEEE Transactions Fuzzy Systems*, 1(1):7–31, 1993.

[182] J. Yin, Y.-X. Wang, and C. Wu. A fuzzy scheduling method in equipment grid. In *Proceedings of the 2007 International Conference on Machine Learning and Cybernetics*, volume 6, pages 3099 –3103, August 2007.

[183] K.F. Man, K.S. Tang, and S. Kwong. Genetic algorithms: concepts and applications [in engineering design]. *IEEE Transactions on Industrial Electronics*, 43(5):519–534, Oct 1996.

[184] B.-D. Liu, C.-Y. Chen, and J.-Y. Tsao. Design of adaptive fuzzy logic controller based on linguistic-hedge concepts and genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 31(1):32–53, Feb 2001.

[185] D. P. Sakas, D. S. Vlachos, and T. E. Simos. Fuzzy neural networks for decision support in negotiation. *AIP Conference Proceedings*, 1060(1):67–70, 2008.

[186] F. Doctor, H. Hagras, V. Callaghan, and A Lopez. An adaptive fuzzy learning mechanism for intelligent agents in ubiquitous computing environments. In *World Automation Congress*, volume 16, pages 101–106, June 2004.

[187] A. M. Acilar and A. Arslan. Optimization of multiple input single output fuzzy membership functions using clonal selection algorithm. In *Proceedings of the 8th Conference on Applied Computer Science*, ACS'08, pages 49–53, Stevens Point, Wisconsin, USA, 2008. World Scientific and Engineering Academy and Society (WSEAS).

[188] K. Hindriks, C. M. Jonker, and D. Tykhonov. The benefits of opponent models in negotiation. In *Proceedings of the IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technologies*, volume 2, pages 439–444, 2009.

[189] P. Huang and K. Sycara. Computational model for online agent negotiation. *Hawaii International Conference on System Sciences*, 1:28b, 2002.

[190] S.S. Fatima, M. Wooldridge, and N. R. Jennings. Multi-issue negotiation with deadlines. *The Journal of Artificial Intelligence Research*, 27:381–417, 2006.

[191] G. Lai, C. Li, and K. Sycara. Efficient multi-attribute negotiation with incomplete information. *Group Decision and Negotiation*, 15(5):511–528, 2006.

[192] S. Adabi, A. M. Movaghar, A. Rahmani, and H. Beigy. Negotiation strategies considering market, time and behavior functions for resource allocation in computational grid. *The Journal of Supercomputing*, 66(3):1350–1389, 2013.

[193] K. M. Sim and C. Y. Choi. Agents that react to changing market situations. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 33(2):188 – 201, apr 2003.

[194] A. Bahrammirzaee, A. Chohra, and K. Madani. An adaptive approach for decision making tactics in automated negotiation. *Applied Intelligence*, 39(3):583–606, 2013.

[195] A. R. Lomuscio, M. Wooldridge, and N. R. Jennings. A classification scheme for negotiation in electronic commerce. *Group Decision and Negotiation*, 12:31–56, 2003.

[196] K. M. Sim. Equilibria, prudent compromises, and the "waiting" game. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 35(4):712 –724, 2005.

[197] T. Baarslag, K. Fujita, E. H. Gerding, K. Hindriks, T. Ito, N. R. Jennings, C. M. Jonker, S. Kraus, R. Lin, V. Robu, and C. R. Williams. Evaluating practical negotiating agents: Results and analysis of the 2011 international competition. *Artificial Intelligence*, 198:73 – 103, 2013.

[198] T. Baarslag, K. Hindriks, C. Jonker, S. Kraus, and R. Lin. The first automated negotiating agents competition (ANAC 2010). In T. Ito, M. Zhang, V. Robu,

S. Fatima, and T. Matsuo, editors, *New Trends in Agent-Based Complex Automated Negotiations*, volume 383 of *Studies in Computational Intelligence*, pages 113–135. Springer Berlin, Heidelberg, 2012.

[199] K. Hindriks, C. M. Jonker, S. Kraus, R. Lin, and D. Tykhonov. Genius: Negotiation environment for heterogeneous agents. In *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems*, volume 2 of *AAMAS '09*, pages 1397–1398, Richland, SC, 2009.

[200] H. Jazayeriy, M. Azmi-Murad, N. Sulaiman, and N. I. Udizir. Pareto-optimal algorithm in bilateral automated negotiation. *International Journal of Digital Content Technology and its Applications*, 5:1–11, 2011.

[201] B. An, K.-M. Sim, L. G. Tang, S. Q. Li, and D.-J. Cheng. Continuous-time negotiation mechanism for software agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 36(6):1261–1272, Dec 2006.

[202] T. Sun, Q. Zhu, S. Li, and M. Zhou. Open, dynamic and continuous one-to-many negotiation system. In *Proceedings of the 2nd International Conference on Bio-Inspired Computing: Theories and Applications*, pages 87–93, Sept 2007.

[203] T. A. Runkler and M. Glesner. Decade - fast centroid approximation defuzzification for real time fuzzy control applications. In *Proceedings of the 1994 ACM Symposium on Applied Computing*, SAC '94, pages 161–165. ACM, 1994.

[204] T.A. Runkler. Extended defuzzification methods and their properties. In *Proceedings of the Fifth IEEE International Conference on Fuzzy Systems*, volume 1, pages 694 –700, September 1996.

[205] A.I. Dounis and C. Caraiscos. Advanced control systems engineering for energy and comfort management in a building environment - a review. *Renewable and Sustainable Energy Reviews*, 13(6-7):1246 – 1261, 2009.

[206] B. Lacroix, C. Paulus, and D. Mercier. Multi-agent control of thermal systems in buildings. In *Proceedings of the 3rd International workshop on Agent Technologies in Energy Systems*, pages n/a–n/a, Valencia, Spain, 2012.

[207] B. Qiao, K. Liu, and C. Guy. A multi-agent system for building control. In *Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, IAT '06, pages 653–659, Washington, DC, USA, 2006. IEEE Computer Society.

[208] The Grid Workload Archive. Available from: http://gwa.ewi.tudelft.nl/, Visited in January 2015.

[209] EGEE Team. LCG. lcg.web.cern.ch/LCG, 2004.

[210] I. Foster, J. Gieraltowski, S. Gose, and et al. The Grid2003 production grid: principles and practice. In *Proceedings of the 13th IEEE International Symposium on High performance Distributed Computing*, pages 236–245, June 2004.

[211] H. Bal, R. Bhoedjang, R. Hofman, and et al. The distributed ASCI supercomputer project. *SIGOPS Operating Systems Review*, 34(4):76–96, October 2000.